

BASIC

BOB ALBRECHT
Le Roy Finkel
JERALD BROWN

Handbuch für Selbststudium und Praxis

ATARI, BASIC wurde aus dem englischen übersetzt und von Dietmar Mayfeld erweitert (siehe Anhang).

Originalfassung ATARI BASIC

Library of Congress Cataloging in Publication Data:

Albrecht, Robert L
Atari BASIC.

(Wiley self-Teaching guides) includes index.

1. Atari 400 (Computer) – Programming.

2. Atari 800 (Computer) – Programming. (Computer program language) I. Finkel, LeRoy, joint outhor. II. Brown, Jerald, 1940 - joint author. III. Title.

ISBN 3–921682–32–0

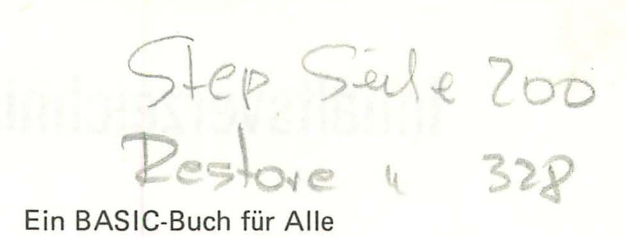
Auflage Herbst 1981
Gedruckt in der Bundesrepublik Deutschland
Printed in West Germany – Imprime'en RFA.

10181

29,80

ATARI®

BASIC



Ein BASIC-Buch für Alle

Hier liegt ein BASIC-Einführungsbuch vor Ihnen, wie Sie es sich schon lange gewünscht haben. Eine leicht verständliche und praktische Einführung mit vielen interessanten Beispielen.

Das Buch ATARI-BASIC eignet sich grundsätzlich für jeden Personalcomputer mit BASIC heute auf dem Markt. Es geht jedoch am Ende des Buches auf die speziellen Farb-, Musik- und Ein-/Ausgabebefehle der ATARI 400 und ATARI 800 Personalcomputer ein.

Der Verfasser Bob Albrecht gehört zu den Pionieren der Personalcomputerbewegung und ist als Lehrer für die Programmsprache BASIC weltweit anerkannt.

Holzkirchen,
Herbst 1981

Winfried Hofacker

Inhaltsverzeichnis

Ein Wort an den Leser

Die Entwicklung integrierter Schaltungen und die Miniaturisierung in der Elektronik haben es ermöglicht, daß man inzwischen einen vollständigen Computer mit allen Peripheriegeräten, bzw. dem erforderlichen Zubehör für weniger als DM 2.000,– kaufen kann. Ein Computer für Anwendungen zu Hause, im Computer-Club, in der Schule oder im Büro liegt damit in der gleichen Preisklasse wie preiswerte Komponenten für Hifi-Anlagen. Diese Computer, unter denen sich auch die Geräte von ATARI befinden, verwenden die Computer-Sprache BASIC. Ziel dieses Buches ist es, Ihnen alle Kenntnisse zu vermitteln, um selbstständig Programme unter Verwendung von BASIC schreiben zu können, und zwar speziell für ATARI-Computer.

BASIC wurde ursprünglich am Dartmouth College von John Kemeny und Thomas Kurtz entwickelt. Sie erkannten die Notwendigkeit für eine Allzweck-Computer-Sprache, die sich besonders für Programmier-Anfänger mit sehr unterschiedlicher Vorbildung eignet. BASIC, ein Kunstwort das entstanden ist aus den Anfangsbuchstaben von "Beginners All-pupose Symbolic Instruction Code", wurde ursprünglich als eine einfache Sprache konzipiert, die sich in wenigen Stunden erlernen ließ. Durch die im Laufe der Jahre erfolgten Verbesserungen kann der Lernvorgang jetzt einige Tage in Anspruch nehmen, aber Sie werden feststellen, daß Sie nahezu alle Probleme und Aufgabenstellungen mit BASIC lösen können.

Wenn Sie schon einen ATARI-Computer zur Verfügung haben, vergewissern Sie sich bitte, daß BASIC bereits in den Speicher des Computers eingesetzt ist, damit er die in BASIC gegebenen Instruktionen versteht. Es liegt in der Natur des Fortschritts, Wege zur Verbesserung bereits vorhandener Dinge zu finden. BASIC stellt in dieser Hinsicht keine Ausnahme dar. Daher werden Sie wahrscheinlich feststellen, daß Ihr BASIC über eine Reihe zusätzlicher Statements (Befehle), sowie einige Weiterentwicklungen verfügt, die in dem 8k-ATARI-BASIC nicht vorhanden waren, das wir beim Schreiben dieses Buches verwendet haben. Daher möchten wir Sie dazu ermutigen, selbst zu experimentieren und, falls Sie einmal nicht weiter kommen sollten, befragen Sie Ihr BASIC-Manual.

Beachten Sie bitte, daß dieses Buch auch über einen Anhang mit BASIC-Funktionen verfügt. Etwa ab Kapitel 3 oder 4 sollten Sie beginnen, im Funktions-Anhang und in Ihrem BASIC-Manual zu blättern, um ein größeres Verständnis für die Möglichkeiten Ihres Computers zu bekommen. Dabei können Sie auch lernen, wie Sie Ihre Programme auf einem Kassetten-Rekorder speichern (CSAVE), sofern Sie ein System mit einem derartigen Gerät gekauft bzw. sich entschlossen haben, Ihren Computer durch einen Rekorder zu erweitern. Sobald Sie nämlich dazu übergehen, längere Programme zu schreiben, die im Laufe der Zeit weiterentwickelt und modifiziert werden, erspart es viel Zeit bei der Eingabe über die Tastatur, wenn man über eine frühere Version des Programms verfügt.

Beachten Sie auch den Anhang mit den Fehler-Mitteilungen. Glauben Sie uns, Sie werden so oft wie wir darauf zurückgreifen um einen Hinweis darauf zu finden, was Sie falsch eingegeben haben oder wo Sie Ihre Logik gelegentlich im Stich gelassen hat. Aber auch in diesem Fall gilt die alte Regel: "Nicht aufgeben, weitermachen!" Daher versuchen Sie es einfach noch einmal.

Jetzt wollen wir uns damit befassen, wie dieses Buch benutzt werden sollte und dann können wir damit beginnen, BASIC zu lernen!

Wie dieses Buch benutzt werden sollte

Dieses Buch ist für den Selbstunterricht geschrieben worden, so daß Sie aktiv in das Erlernen von ATARI BASIC einbezogen werden. Der Stoff ist in kurze, numerierte Abschnitte unterteilt, die als "Frames" bezeichnet werden. Jeder dieser Abschnitte vermittelt Ihnen irgendeine neue Erkenntnis über ATARI BASIC und stellt Ihnen eine Frage, bzw. fordert Sie auf, ein Programm zu schreiben. Die jeweils korrekten Antworten stehen unterhalb der gestrichelten Linien. Für ein möglichst effektives Lernen empfehlen wir Ihnen, Sie mit einem Stück Karton solange abzudecken, bis Sie Ihre eigene Antwort geschrieben haben.

Sie werden am besten lernen, wenn Sie die Antworten tatsächlich hinschreiben und die Programme auf Ihrem ATARI-Computer ausprobieren. Die Fragen sind sehr sorgfältig unter dem Gesichtspunkt ausgewählt worden, Ihre Aufmerksamkeit auf wichtige Punkte in den

Beispielen und Erklärungen zu lenken und Ihnen dabei zu helfen, das anzuwenden, was erklärt oder demonstriert wurde.

Jedes Kapitel beginnt mit einer Liste der Lernziele, die Ihnen angibt, was Sie nach Durcharbeiten dieses Kapitels dazugelernt haben. Sofern Sie bereits gewisse Vorkenntnisse in BASIC haben und Ihnen die Lernziele für dieses Kapitel vertraut erscheinen, arbeiten Sie zunächst den Eigentest am Ende des entsprechenden Kapitels durch, um festzustellen, wo Sie mit dem Lesen des Buches beginnen sollten. Ist Ihr Test gut ausgefallen, dann beschäftigen Sie sich nur mit den Abschnitten, die zu den Fragen angegeben sind, die Sie nicht lösen konnten. Sofern Sie viele Fragen nicht beantworten konnten, beginnen Sie mit der Arbeit am Anfang dieses Kapitels.

Der Eigentest kann auch zur Wiederholung des im vorangegangenen Kapitels vermittelten Stoffes verwendet werden. Sie können Ihre Kenntnisse direkt nach Abschluß des Kapitels prüfen. Oder aber Sie machen nach dem Lesen eines Kapitels eine Pause und benutzen den Eigentest zur Wiederholung, bevor Sie mit dem nächsten Kapitel beginnen.

Am Endes des Buches ist ein Abschluß-Eigentest vorgesehen, der es Ihnen ermöglicht, Ihre gesamten Kenntnisse in ATARI BASIC zu überprüfen.

Dieses Buch reicht zwar völlig zum Erlernen von ATARI BASIC aus, aber alles was Sie lernen wird solange blanke Theorie bleiben, bis Sie tatsächlich selbst an einem Computer-Terminal sitzen und Ihre Kenntnisse der Computer-Sprache und der Programmier-Technik anwenden. Daher möchten wir Ihnen unbedingt empfehlen, dieses Buch zusammen mit einem ATARI Computer durchzuarbeiten. BASIC wird leichter zu erlernen und verständlicher, - auch wenn Sie nur gelegentlich Zugang zu einem Computer haben - , da Sie dann die Beispiele und Übungen ausprobieren, Modifikationen durchführen und Ihre eigenen Programme entwickeln können. Jetzt sind Sie bereit, sich selbst beizubringen, wie man ATARI BASIC nutzt.

NOTIZEN

Kapitel 1

Ihr ATARI Personal-Computer

Dieses Kapitel soll Sie mit den für Anwendungen zu Hause, im Computer-Club, in der Schule oder im Büro bzw. Labor geeigneten Personal-Computern ATARI 400 und ATARI 800 bekannt machen. Wenn Sie dieses Kapitel beendet haben, werden Sie ausreichende Grundkenntnisse über diese beiden Computer und die für Sie zur Zeit erhältlichen Erweiterungs-Komponenten haben. Ebenso werden Sie in der Lage sein, die folgenden Worte und Ausdrücke aus der Sprache der Computer-Welt zu benutzen:

Computer-Sprache
BASIC und speziell ATARI BASIC
CPU oder Zentraleinheit
Keyboard oder Tastatur
Bildschirm
Drucker
Programm
Programm-Rekorder
ROM-Kassette mit ATARI BASIC
Betriebs-System
ROM-Modul
RAM-Speicher
RAM-Speicher-Modul
BASIC-Statements
Zeilennummern

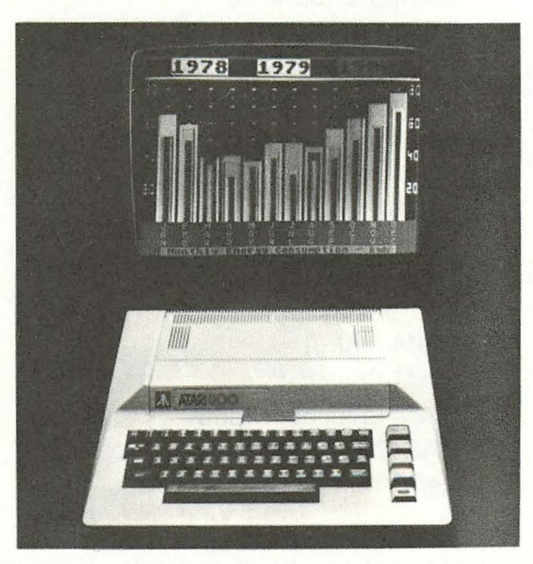
Sie können dieses Buch entweder zusammen mit dem ATARI 400 oder dem ATARI 800 Personal-Computer-System benutzen. Der Typ 400 besteht aus der Computer-Konsole 400, einem Keyboard mit Sensortasten, Ihrem Fernsehgerät und möglicherweise zum zusätzlichen Programm-Rekorder ATARI 410. Das Modell 800 verfügt über eine Schreibmaschinen-ähnliche Tastatur und wird komplett mit dem Kassetten-Rekorder geliefert. Zwischen beiden Systemen bestehen noch weitere Unterschiede, die von Fall zu Fall erläutert werden sollen.



ATARI 400



ATARI 800



Bevor Sie weiterlesen wäre es zweckmäßig, Ihren Computer so aufzubauen, daß Sie ihn während des Lesens benutzen können. In dem Karton, in dem Ihr ATARI 400 oder 800 geliefert wurde, befindet sich auch die zugehörige Bedienungsanleitung. Diese Broschüre erläutert ausführlich, wie Sie Ihren Fernseher anschließen müssen, die Spannung für den Computer einschalten und die mitgelieferten Programm-

Kassetten verwenden können. Das Bedienungs-Handbuch enthält auf wenigen Seiten eine Menge an Informationen, jedoch müssen Sie keineswegs alles durchlesen. Allerdings sollten Sie Ihr System soweit vorbereiten, daß es eingeschaltet und in BASIC arbeitsfähig ist, sofern Sie das nicht bereits getan haben. Anschließend kehren Sie bitte wieder zur Lektüre dieses Buches zurück!

Schön, daß Sie wieder da sind. Wir wollen uns jetzt etwas näher mit Ihrem ATARI Computer vertraut machen. Verständlicherweise zieht das Keyboard als erstes die Aufmerksamkeit auf sich.



Das Tastenfeld bietet Ihnen die Möglichkeit, mit Ihrem ATARI Computer in Verbindung zu treten. Es wird als Eingabevorrichtung bezeichnet, da Sie es dazu benutzen können, Informationen in den Computer einzugeben.

1. Das Keyboard ist die Eingabevorrichtung, die es Ihnen ermöglicht, mit dem Computer in der Computer-Sprache .... in Verbindung zu treten.

BASIC

2. Derjenige Teil der Computer-Elektronik, der tatsächlich "rechnet", wird als CPU oder "Zentraleinheit" bezeichnet. Er ist auf einer einzelnen gedruckten Schaltung unterhalb der geriffelten Abdeckung an der Rückseite der Konsole untergebracht.

Ebenfalls unter dieser Abdeckung befindet sich der Speicher des Computers. Der erste Abschnitt dieser Speicher-Bank umfaßt den ROM-

Modul (ROM = "Nur-Lese-Speicher") mit dem 8 k-Betriebssystem. Der ROM-Modul enthält ein Programm, eine Folge von Instruktionen, die dem Computer vorschreiben, wie er auf Mitteilungen von der Außenwelt zu antworten hat. Für uns sind das ROM und die darin enthaltenen Programme Bestandteile des Computers selbst.

a) Rechenvorgänge erfolgen in der ....

b) Das Betriebs-System ist in einem anderen Teil der Elektronik gespeichert, der die Bezeichnung ROM trägt. Diese Abkürzung bedeutet:

....

- -

a) CPU (Zentraleinheit)
b) Nur-Lese-Speicher

3. Wir haben bereits das "8 k-Betriebssystem" kennengelernt, das dem Computer hilft, unsere an ihn gerichteten Instruktionen zu interpretieren. Die Bezeichnung "8 K" stellt eine übliche Methode zur Angabe der Größe eines Computer-Speichers dar. Der Buchstabe K steht dabei für jeweils 1.000 Speicher-Bytes. "Byte" ist ein weiterer Ausdruck aus der Computer-Sprache und bezeichnet eine einzelne Speicher-Einheit. Jedes Speicher-Byte kann ungefähr einen Buchstaben, eine Ziffer oder ein anderes Zeichen speichern, so daß 8 K ungefähr 8000 Zeichen aufnehmen können.

16 K bedeutet daher eine Kapazität zum Speichern von ungefähr .... Zeichen.

- -

16.000

4. Sowohl das Modell 400 als auch der 800 enthalten einen zweiten Speicher-Abschnitt mit 8 K RAM oder "Schreib-Lese-Speicher". In diesem Bereich speichert der Computer Informationen, die Sie über die Tastatur eingeben. Wir bezeichnen den RAM-Speicher oft auch als "Arbeitsspeicher". Wir können Informationen in ihn schreiben, oder sie mit wenigen Tastenbetätigungen löschen.

Der RAM-Speicher unterscheidet sich vom ROM, dem "Nur-Lese-

Speicher". Der Computer kann den Inhalt des ROM's lesen, der ständig erhalten bleibt. Er kann jedoch von der Tastatur aus weder gelöscht noch verändert werden.

Beantworten Sie bitte die folgenden Fragen, indem Sie entweder RAM, ROM oder beides einsetzen.

a) Welcher Speicher kann gelöscht werden? ....

b) Welcher Speicher kann nicht geändert werden? ....

c) Welcher Speicher hat eine mit "8 K" bezeichnete Kapazität ....

d) Welcher Speicher nimmt die über die Tastatur eingebebenen Informationen auf? ....

---

a) RAM
b) ROM
c) RAM und ROM
d) RAM

5. Wenn Sie einen ATARI 800 besitzen, können Sie weitere RAM-Speicher-Module hinzukaufen, um den Arbeitsspeicher Ihres Computers zu erweitern. Bei der Benutzung dieses Buches benötigen Sie keinen zusätzlichen Speicher, aber später werden Sie vermutlich längere Programme schreiben wollen, die mehr RAM zur Unterbringung benötigen.

Vergewissern Sie sich, daß Sie die ATARI-BASIC-Kassette in den zugehörigen Schacht Ihres Computers eingesteckt haben. (Folgen Sie dabei der Instruktionen in Ihrer Bedienungsanleitung.) Das Einsetzen der Kassette bedeutet für den Computer eine Vergrößerung seines Speichers und gibt ihm zusätzliche Informationen. Diese Kassette enthält ein weiteres vorprogrammiertes 8 K-ROM; unmittelbar nach dem Einsetzen der Kassette "weiß" der Computer, wie ATARI BASIC "gesprochen" wird. Jetzt müssen Sie ATARI BASIC erlernen, um dem Computer Instruktionen geben zu können. Eine Folge von einer oder mehr Instruktionen, die dem Computer sagt, was er tun soll, wird als Programm bezeichnet. Ihr Programm, bzw. Ihre Instruktionen müssen über das Keyboard in BASIC eingetastet werden und zwar gemäß den

Regeln von BASIC, die Sie in diesem Buch erlernen werden. Ihre Instruktionen, die dem Computer sagen, was er nach Ihrem Wunsch und gemäß den Regeln von BASIC tun soll, werden als .... bezeichnet.

Programm

6. Nehmen wir einmal an, Sie tippen ein Programm in den Computer ein. Sobald der Computer von der Tastatur ein Zeichen erhält (wenn Sie mit dem Schreiben beginnen), bringt er das gleiche Zeichen auf dem Bildschirm zur Anzeige. Er gibt uns damit eine Darstellung der von uns eingegebenen Zeichen.

Der Bildschirm des Fernsehgerätes wird daher als Ausgabe-Gerät bezeichnet, da er empfängt, was wir eingeben und es unmittelbar wieder ausgibt. Das gibt Ihnen die Gewißheit, daß Sie auch wirklich das eingegeben haben, was Sie vorhatten. Wenn später der Computer den Instruktionen in Ihrem Programmen folgt, werden die Ergebnisse ebenfalls auf dem Bildschirm angezeigt.

Dies ist ein weiteres Beispiel für einen Heim-Fernseher als .... - Gerät.

Ausgabe

7. Sehen Sie sich nochmals das Bild des an einen Fernseher angeschlossenen ATARI-Computers 400 an.

a) Wir verwenden die Tastatur zur Eingabe von Informationen in den Computer. Was geschieht mit den Informationen, die wir eintippen?
....
....

b) Wie tritt der Computer mit uns, den Anwendern, in Verbindung?
....
....

---

a) Sie werden an den Computer übermittelt und anschließend auf dem

Bildschirm dargestellt.
b) Er bildet die Informationen auf dem Bildschirm ab.

8. Wenn Sie einen ATARI 800 haben, verfügen Sie auch über einen Programm-Rekorder, den Sie in Verbindung mit Ihrem Computer benutzen können. Er ist einem Heim-Kassetten-Rekorder ähnlich. (Sie können den Kassetten-Rekorder auch zusätzlich zu Ihrem System 400 anschaffen.)

Der Programm-Rekorder ist ein völlig vom elektronischen Computer-Speicher getrenntes Speicher-Gerät. Er kann zum Aufzeichnen Ihrer Computer-Programme und von Informationen (die als Daten bezeichnet werden) wie Adreßlisten, Rezepten, Terminen, Zahlen aus Ihrem häuslichen Finanzplan oder Ihrem Kontostand verwendet werden. Sobald Sie die BASIC-Computer-Sprache gemeistert haben, werden Sie feststellen, wie handlich und leicht zu bedienen dieses Speichergerät ist. Neben Magnetband-Kassetten gibt es auch noch andere Möglichkeiten, Informationen und Programme außerhalb des Computers zu speichern.

Alle derartigen externen Speicherverfahren bieten eine bequeme Möglichkeit, Programme und Informationen, die oft benötigt werden, in den Computer einzuspeisen oder zu "laden", ohne daß jedesmal der nicht unbeträchtliche Zeitaufwand zum Eintippen über die Tastatur erforderlich wird. Magnetische Speicherplatten sind ein weiteres externes Speicher-Medium. In diesem Buch werden jedoch nur die drei Haupt-Komponenten eines Computer-Systems benötigt, die wir zuvor besprochen haben. Dabei handelt es sich um ....
....

– –

einen Computer, eine Tastatur und einen Fernseh-Bildschirm.

9. Rufen Sie sich noch einmal ins Gedächtnis zurück, was Sie in den Abschnitten 1 – 8 gelesen haben. Wir haben darin den ATARI-Computer beschrieben. Für unsere Zwecke sind seine wichtigsten Komponenten:

1) Der Computer selbst - wobei die CPU mit der zugehörigen Elektronik in der Konsole untergebracht ist.

2) Ein Eingabegerät - die Tastatur, die wir dazu benutzen, Informationen in den Computer einzugeben.

3) Ein Ausgabegerät - der Fernseh-Bildschirm, der Informationen darstellt, die über das Keyboard eingegeben oder vom Computer übermittelt wurden.

4) BASIC - betriebsbereit im Computer installiert

In diesem Buch werden wir Ihnen viele Programme in ATARI BASIC vorstellen und Ihnen helfen, das Lesen, Verstehen und Nutzen dieser Programme zu Ihrem eigenen Vergnügen zu erlernen. Wir werden uns dabei auf Anwendungen konzentrieren, von denen wir glauben, daß sie für Benutzer von Heim-, Schul- oder Personal-Computern von Interesse sein dürften. Daher stellen wir Ihnen jetzt, sozusagen zur Appetitanregung, ein Computer-Spiel vor.

Die folgenden Zeilen, bis zum Wort RUN, sind ein Computer-Programm.

```
100 REMARK *** DIES IST EIN EINFACHES COMPUTER-SPIEL
110 LET X = INT(100* RND(1))+1
120 PRINT
130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100."
140 PRINT "RATEN SIE MEINE ZAHL!!!"
150 PRINT : PRINT "IHR VERSUCH"; : INPUT G
160 IF G < X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL." : GOTO 150
170 IF G > X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    KLEINEREN ZAHL." : GOTO 150
180 IF G=X THEN PRINT "GRATULIERE!!! SIE HABEN MEINE
    ZAHL ERRATEN." : GOTO 110

RUN
```

Was Sie auf dem Bildschirm sehen können, lesen Sie anschließend. Die jeweiligen, vom Benutzer geratenen Zahlen, müssen über die Tastatur eingegeben werden.

```
ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100.
RATEN SIE MEINE ZAHL!!!

IHR VERSUCH? 50
VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL.

IHR VERSUCH? 75
VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL.

IHR VERSUCH? 65
VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL.

IHR VERSUCH? 58
VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL.

IHR VERSUCH? 62
VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL.

IHR VERSUCH? 60
VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL.

IHR VERSUCH? 61
GRATULIERE!!! SIE HABEN MEINE ZAHL ERRATEN.

ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100.
RATEN SIE MEINE ZAHL!!!

IHR VERSUCH ?
```

usw.

Diese Zeilen werden als der "Lauf" oder "Run" des Programms bezeichnet. Der Computer erzeugt dazu eine Zufallszahl zwischen 1 und 100. Der Spieler muß seine Zahlen eintippen. Nach jedem Versuch gibt der Computer einen Hinweis, um dem Spieler beim Raten der Zahl zu helfen.

Sofern Ihnen das jetzt ein bißchen verwirrend vorkommt, lesen Sie einfach weiter. Alles wird noch ausführlich erläutert, und es wird gar nicht lange dauern, bis Sie selbst Programme in BASIC, wie dieses Beispiel lesen und schreiben können.

Jeweils nach dem Erraten der Computer-Zahl beginnt der Computer mit einem neuen Spiel, wobei es sich mit einiger Wahrscheinlichkeit um eine andere Zahl handeln wird. Sie können dieses Spiel so lange fortsetzen, wie Sie wollen.

Sehen Sie sich jetzt noch einmal das Programm für unser Computer-Spiel an. Es besteht aus neun Zeilen, von denen jede ein oder mehrere BASIC-Statements enthält. Jedes Statement beginnt mit einer Zeilennummer, z. B. 130 in der vierten Zeile. Der darauffolgende Text, in Zeile 130 also PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 und 100" ist ein Statement.

In unserem Programm enthält jede numerierte Zeile ein oder mehrere BASIC- . . . . Die Zahlen 100 bis 180 werden als . . . . bezeichnet.

— —

Statements; Zeilen-Nummern

10. Das Programm in Abschnitt 9 wurde Zeile für Zeile über die Tastatur eingetippt. Beim Schreiben wurde das Programm im Speicher des Computers abgelegt und gleichzeitig auf dem . . . . dargestellt.

— —

Bildschirm

Bitte beachten Sie: Auf dem Fernsehschirm können bis zu 24 Zeilen gleichzeitig abgebildet werden. Sobald der Schirm gefüllt ist, verursacht die Eingabe weiterer Informationen das "Aufwärtsrollen" des Bildschirminhaltes, wodurch die obersten Zeilen nacheinander über den Rand weggeschoben werden und verschwinden.

11. Zuerst haben wir das ganze Programm (Zeilen 100 bis 180) eingetippt und am Ende jeder Zeile die Taste RETURN gedrückt. Dieser Vorgang wird als "Eingabe des Programms" bezeichnet. Er bewirkt, daß das Programm im Computer-Speicher abgelegt wird. Anschließend haben wir RUN eingegeben. Dadurch erhielt der Computer die Anweisung, das Programm auszuführen. Mit anderen Worten: Nach dem Speichern des

Programms haben wir dem Computer gesagt, er soll den Instruktionen (Statements) des Programms folgen und damit das Programm ausführen.

Befindet sich also ein Programm im Speicher des Computers, wird er durch Eingabe von RUN dazu veranlaßt, ...
....

– –

die Instruktionen des Programms auszuführen.

12. Während des Laufs gehorchte der Computer den Instruktionen (Statements) folgendermaßen: Zuerst erzeugte er eine Zufallszahl zwischen 1 und 100 einschließlich (Zeile 110). Diese Zahl ist ganzzahlig ("Integer"), d. h. sie hat keinen Dezimalbruch-Anteil.

```
110 LET X = INT(100*RND(1))+1
```

Als nächstes gibt der Computer dem Spieler Anweisungen (Zeilen 120, 130, 140 und 150).

```
100 REMARK *** DIES IST EIN EINFACHES COMPUTER-SPIEL
120 PRINT
130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100",
140 PRINT "RATEN SIE MEINE ZAHL!!!"
```

Dann forderte der Computer den Spieler auf, einen Versuch zu machen (Zeile 150).

```
150 PRINT : PRINT "IHR VERSUCH" ; INPUT G
```

Nachdem der Spieler eine Zahl eingetippt und die Taste RETURN betätigt hatte, verglich der Computer die geratene Zahl mit seiner dem Spieler unbekannten Zahl und gab ihm die entsprechende Antwort (Zeilen 160, 170, 180).

```
160 IF G < X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL." : GOTO 150
170 IF G > X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    KLEINEREN ZAHL." : GOTO 150
```

```
180 IF G=X THEN PRINT "GRATULIERE!!! SIE HABEN MEINE
    ZAHL ERRATEN." : GOTO 110
```

Wenn der Spieler die Zahl des Computers nicht erriet, ging der Computer im Programm zurück zur Zeile 150 und forderte zu einem neuen Versuch auf. Sofern der Spieler jedoch die unbekannte Zahl traf, bestätigte der Computer die korrekte Eingabe und ging im Programm zurück zur Zeile 110, um sich eine neue Zahl "auszudenken".

```
100 REMARK *** DIES IST EIN EINFACHES COMPUTER-SPIEL
```

Jetzt haben wir nur noch die Zeile 100 vergessen. REMARK ist auch ein Statement. Es fordert jedoch nicht den Computer auf, irgend etwas zu tun, sondern ist nur vorgesehen, um dem Leser des Programms etwas über das Programm selbst mitzuteilen.

Welches Wort müssen wir über die Tastatur eingeben, um dem Computer mitzuteilen, daß er ein Programm ausführen soll? ....

RUN

Wenn Sie jetzt zum Schluß festellen wollen, wieviel Sie aus diesem ersten Kapitel gelernt haben, versuchen Sie noch einmal den anschließenden Eigentest. Dann geht es weiter mit Kapitel 2, in dem Sie anfangen werden, die tatsächliche Benutzung des Computers zu erlernen.

EIGENTEST

Versuchen Sie diesen Eigentest, um festzustellen, wieviel Sie bisher gelernt haben.

1. Was benutzt Ihr Computer als Ausgabegerät? ....

2. Wofür stehen die folgenden Abkürzungen?
   a) CPU ....
   b) RAM ....
   c) ROM ....

3. Ein Computer hat ja bekanntlich keine Beine. Was wollen wir also von ihm, wenn wir das Wort RUN (Lauf) eingeben? ....
   ....

Antworten zum Eigentest

Die Zahlen in den Klammern hinter den jeweiligen Antworten beziehen sich auf diejenigen Abschnitte des ersten Kapitels, in denen der entsprechende Stoff behandelt wurde. Wenn Sie sich nochmals kurz informieren möchten, können Sie dort nachschlagen.

1. Fernseh-Bildschirm (6)
2. a) Zentraleinheit
   b) Schreib/Lesespeicher oder Arbeitsspeicher
   c) Nur-Lese-Speicher (2, 4)
3. Führe das Programm aus (11, 12)

NOTIZEN

Kapitel 2

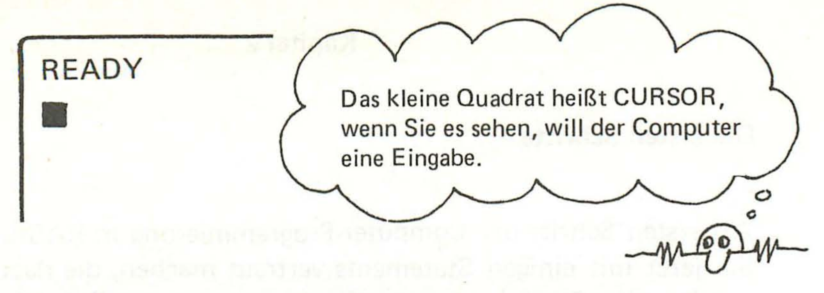
Die ersten Schritte

Als ersten Schritt der Computer-Programmierung in BASIC wollen wir Sie jetzt mit einigen Statements vertraut machen, die dazu verwendet werden, den Computer zu instruieren, was er tun soll. In diesem Kapitel werden Sie zunächst den direkten oder unmittelbaren Betrieb verwenden. Dabei teilen Sie dem Computer mit, was er ausführen soll, und er erledigt es unmittelbar darauf. Wenn Sie dieses Kapitel beendet haben, werden Sie in der Lage sein:

- direkte Statements zur Instruktion des Computers zu benutzen;
- Fehlermeldungen des Computers zu erkennen;
- das PRINT-Statement mit Anführungszeichen zu verwenden, um Strings (Mitteilungen) auszudrucken;
- Tippfehler zu korrigieren oder ein fehlerhaftes Statement zu entfernen;
- direkte Statements für arithmetische Operationen zu benutzen;
- Werte einfacher mathematischer Ausdrücke, unter Verwendung der für die Arithmetik gültigen Symbole und Regeln von BASIC, zu berechnen;
- Gleitkomma- bzw. Exponential-Darstellungen von Zahlen zu erkennen und sie in gewöhnliche Zahlen umzuwandeln.

1.
Jetzt wollen wir damit beginnen, mit unserem Computer "zu sprechen". Dabei gehen wir davon aus, daß Ihr ATARI-Computer-System betriebsbereit und die BASIC-Sprach-Kassette eingesetzt ist. Anweisungen dafür finden Sie im Bedienungs-Handbuch Ihres Computers.

Fertig? Dann Schalten Sie jetzt Ihren Computer ein. KLICK! Auf Ihrem Bildschirm sehen Sie foglende Darstellung:

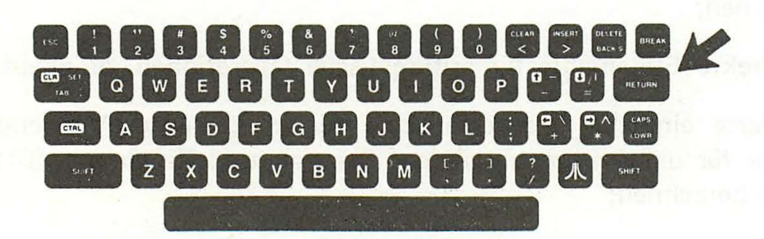


Das kleine helle Rechteck wird als CURSOR bezeichnet. Wenn Sie ihn sehen, wissen Sie, daß Sie an der Reihe sind, etwas zu tun.

Ihr ATARI-Computer ist jetzt für Sie zur Benutzung bereit. Daß er bereit ist wissen Sie aus dem Wort READY auf dem Bildschirm und aus der Darstellung des kleinen Rechteckes, das als .... bezeichnet wird.

CURSOR

2. Um mit dem Computer in Verbindung zu treten verwenden wir natürlich die Tastatur.



In obiger Abbildung zeigt ein dicker, schwarzer Pfeil auf eine Taste, die bezeichnet ist mit ....

RETURN

3. Ihr Computer wartet geduldig darauf, daß Sie etwas tun. Geben Sie daher doch versuchsweise einmal Ihren Namen ein und drücken Sie die RETURN-Taste. Der Computer wird vermutlich eine Fehlermeldung auf dem Bildschirm ausgeben.

Hier sehen Sie, was geschah, als Karl seinen Namen eintippte.

```
KARL ■
```

Während Karl schrieb, bewegte sich der Cursor nach rechts. Zum Schluß betätigte Karl die RETURN-Taste, worauf auf dem Bildschirm folgende Anzeige sichtbar wurde:

```
KARL
ERROR– KARL ■
```

Der Computer gibt an, daß ein Fehler (ERROR) vorliegt, wobei es sich offensichtlich um das Wort KARL handelt, das durch ein kleines Rechteck gekennzeichnet ist. Anschließend schaltet er den Cursor wieder ein.

Warum ist dies wohl geschehen? Wir sehen an diesem Beispiel ganz deutlich, daß der Computer nur über eine sehr beschränkte Intelligenz verfügt. Er hat das Wort KARL einfach nicht verstanden!

a) Wenn Sie etwas eintippen, auf RETURN drücken und der Computer auf dem Bildschirm ERROR meldet, was will er Ihnen damit sagen?
....

b) Was macht der Computer, nachdem er "ERROR-- KARL " ausgegeben hat? ....

– –

a) Er versteht Sie nicht.
b) Er schaltet den Cursor wieder ein.

Mit dem Cursor teilt der Computer Ihnen mit, daß alles in Ordnung ist. Er ist nämlich sehr geduldig und vergibt Fehler sehr schnell.

Wenn Sie einen Fehler machen, bringt der Computer ihn zur Anzeige und schaltet anschließend den Cursor wieder an, um Ihnen mitzuteilen, daß er für einen erneuten Versuch von Ihnen bereit ist.

4. Suchen Sie auf der Tastatur die SHIFT- und die CLEAR-Taste. Drücken Sie die SHIFT-Taste, halten Sie sie fest und drücken Sie gleichzeitig auf CLEAR. Der Bildschirm ist dann völlig leer, bis auf den CURSOR. Das Drücken von SHIFT und CLEAR gemeinsam löscht also den gesamten Bildschirm und läßt nur den Cursor in seiner "Home"-Position zurück. Darunter versteht man die Ausgangsposition in der linken oberen Ecke des Bildschirms.

a) Um den Bildschirm zu löschen und den Cursor in die Home-Position zu bringen, müssen Sie .... und .... gleichzeitig drücken.

b) Wo befindet sich die Home-Position des Cursors? ....
....

---

a) SHIFT; CLEAR
b) In der linken oberen Ecke des Bildschirms.

Eine andere Möglichkeit, um den Bildschirm zu löschen und den Cursor in die Home-Position zu bringen besteht darin, CTRL und CLEAR gemeinsam zu drücken. CTRL ist eine Abkürzung für CONTROL. Die CTRL-Taste liegt an der linken Seite der Tastatur, direkt oberhalb der SHIFT-Taste.

5. Um Mißverständnisse mit einem Computer zu vermeiden, müssen wir seine Sprache erlernen. Wir wollen daher jetzt mit einigen einfachen einzeiligen Statements beginnen, die der Computer versteht.

In diesem Kapitel werden wir direkte Statements verwenden. Direkte Statements haben keine Zeilennummern. Wenn Sie ein direktes State-

ment eingeben und RETURN drücken, führt der Computer das Statement direkt aus und vergißt es dann. Wir bezeichnen dies als "direkte BASIC-Betriebsart". Hier sehen Sie ein weiteres Beispiel für ein Statement, das im direkten Betrieb ausgeführt wird.

Wir geben über die Tastatur ein:

PRINT "MEIN BENUTZER VERSTEHT MICH"

Dann drücken wir auf RETURN.

Der Computer schreibt auf dem Bildschirm:

MEIN BENUTZER VERSTEHT MICH

Vervollständigen Sie jetzt selbst das nächste Beispiel:

Wir schreiben:

PRINT "EINBRECHER SIND IM HAUS!"

Der Computer schreibt auf dem Bildschirm:

....

EINBRECHER SIND IM HAUS!

Zusammen mit der Meldung, daß Einbrecher im Hause sind, könnten wir auch vorsehen (wie wir später noch sehen werden), daß der Computer ein akustisches Alarm-Signal abgibt. Beispielsweise könnte er den Jailhouse-Blues spielen! Das würde den Einbrecher vielleicht an die möglichen Folgen einer Straftat erninnern und ihn daher von dem geplanten Diebstahl abschrecken. Mehr darüber jedoch später.

6. Das Statement PRINT "MEIN BENUTZER VERSTEHT MICH" wird als PRINT-Statement bezeichnet. Es weist den Computer an, etwas auf den Bildschirm zu schreiben. Ausgegeben wird die Mitteilung, die auf das Wort PRINT folgt. Bitte beachten Sie, daß dieser Text in Anführungsstriche gesetzt ist. Eine derart gekennzeichnete Mitteilung wird als "String" bezeichnet.

In unserem Beispiel ist der Text MEIN BENUTZER VERSTEHT MICH

der eigentliche String, die ihn einschließenden Anführungszeichen gehören nicht dazu.

Ein String kann enthalten:

Ziffern (0, 1, 2, . . .)
Buchstaben (A, B, C, . . .)
Spezielle Zeichen (+, –, *, /, Komma, usw.)

Glauben Sie daß Anführungszeichen, die ja Anfang und Ende eines Strings kennzeichnen, auch innerhalb eines Strings als Zeichen verwendet werden dürfen ? . . . .
. . . .
. . . .

– –

Nein, Sie dürfen nicht verwendet werden, da dies den Computer wirklich in Verwirring bringen würde. Dagegen können Apostrophe (') benutzt werden, wie das folgende Beispiel angibt:

Wir geben ein:

PRINT "SIE SAGTEN, 'ES IST ALLES IN ORDNUNG.' "

Der Computer schreibt auf den Bildschirm:

SIE SAGTEN, 'ES IST ALLES IN ORDNUNG.'

Bitte vervollständigen Sie das folgende Statement, damit der Computer das schreibt, was wir als Ausgabe auf dem Bildschirm lesen wollen:

Wir geben ein:

PRINT . . . .

Auf dem Bildschirm soll zu lesen sein:

DER BRATEN IST FERTIG. SCHALTE DEN OFEN AUS.

– –

"DER BRATEN IST FERTIG. SCHALTE DEN OFEN AUS."

Haben Sie sich an die Anführungszeichen erinnert? Eines Tages wird

der Computer sicherlich den Ofen ausschalten, denn nahezu jedes elektrische Gerät kann, sofern es über geeignete Anschlüsse verfügt, von einem Computer mit dem richtigen Programm gesteuert werden.

7. Haben Sie bisher schon irgendwelche Tippfehler gemacht? Sollte Ihnen das passieren, gibt es in BASIC eine ganz einfache Möglichkeit zu Ihrer Korrektur. Achten Sie einmal darauf, was geschieht, wenn wir einen Tippfehler machen.

Wir geben beispielsweise über die Tastatur ein:

PTINT "DENTIST APPOINTMENT TODAY"

Der Computer schreibt auf den Bildschirm:

ERROR – PTINT "DENTIST APPOINTMENT TODAY"

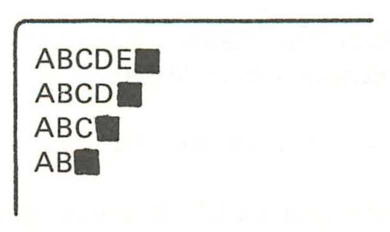
Wir haben PRINT falsch geschrieben, so daß der Computer nicht weiß, was wir wollen. Hätten wir jedoch gleich beim Schreiben bemerkt, daß wir T statt R getippt haben, wäre es möglich gewesen, den Fehler sofort mit der "DELETE BACK S" - Taste zu beseitigen. Jedesmal beim Betätigen dieser Taste bewegt sich der Cursor um eine Stelle nach links und löscht dabei das an dieser Stelle stehende Zeichen.

Wir schreiben ABCDE

Betätigen Sie jetzt die Taste DELETE BACK S.

Drücken Sie noch einmal auf DELETE BACK S!

Und noch einmal!



■ = CURSOR

Was geschieht, wenn wir DELETE BACK S noch zweimal betätigen? . . .

. . . .

– –

Der Cursor bewegt sich zwei Stellen nach links, wodurch E, D und C gelöscht werden.

8. Vervollständigen Sie die folgende Zeile so, daß der Computer den Text in der darunterstehenden Zeile druckt.

Wir schreiben: PRIMR .... "MORGEN IST DER GEBURTSTAG DEINER MUTTER!!!"

Der Computer gibt auf dem Bildschirm aus: "MORGEN IST DER GEBURTSTAG DEINER MUTTER!!!

– –

| DELETE BACK S | | DELETE BACK S | NT

Wir erinnern uns: Um das zuletzt geschriebene Zeichen zu löschen, betätigen wir DELETE BACK S. Um zwei Zeichen zu löschen, drücken wir zweimal auf DELETE BACK S usw.

9. Die Taste DELETE BACK S ist hervorragend geeignet, um Fehler zu beseitigen, die man gerade eben gemacht hat. Aber nehmen wir einmal an, wir schreiben eine lange Zeile, sind fast an ihrem Ende angelangt und entecken dann zu unserem Ärger einen Fehler ganz am Anfang. Vervollständigen wir die Zeile ohne Korrektur und drücken RETURN, werden wir mit einiger Wahrscheinlichkeit eine Fehlermeldung erhalten. Sie werden also sicherlich die Zeile entfernen, das heißt von Anfang an löschen wollen.

Das geht auch ganz einfach. Sie brauchen dazu nur die SHIFT-Taste und DELETE BACK S gleichzeitig zu betätigen.

Wir schreiben: PRIMT "ES WAR EINMAL EIN

Jetzt betätigen wir SHIFT und DELETE BACK S zusammen. Der Computer löscht die Zeile und schaltet den Cursor ein. Was will der Computer uns mitteilen, wenn er den Cursor sichtbar macht?

....

– –

Wir sind an der Reihe zu schreiben.

10. Wir können dem Computer auch den Befehl geben, arithmetische Operationen auszuführen und die Antwort auszugeben. Mit anderen Worten, wir können den Computer auch als Tischrechner benutzen.

Wir schreiben PRINT 7 + 5

Er schreibt: 12

Das Statement PRINT 7 + 5 gibt dem Computer den Befehl, den Wert von 7 + 5 zu berechnen und das Ergebnis anzuzeigen.

Bitte beachten Sei, daß tatsächlich auszuführende Berechnungen für den Computer dadurch gekennzeichnet sind, daß keine Anführungszeichen im PRINT-Statement verwendet werden. Vergleichen Sie die nachstehend angegebenen Formate:

1.) PRINT "13 + 6"

Die in Anführungszeichen eingeschlossenen Zahlen stellen einen String dar. Der Computer gibt ihn auf dem Bildschirm in der gleichen Form wieder, wie er eingegeben wurde: 13 + 6

2.) PRINT 13 + 6

Dies ist ein numerischer Ausdruck (ohne Anführungszeichen). Der Computer führt die geforderte arithmetische Operation durch und gibt das Ergebnis aus: 19

Jetzt sind Sie an der Reihe. Geben Sie in den nächsten Zeilen jeweils an, was der Computer auf unsere Eingabe hin auf dem Bildschirm ausgibt.

Wir schreiben: PRINT 23 + 45

Der Computer schreibt: . . . .

Wir schreiben: PRINT 1 + 2 + 3 + 4 + 5

Der Computer schreibt: . . . .

Wir schreiben: PRINT "1 + 2 + 3 + 4 + 5"

Der Computer schreibt: ....

— —

68

15

1 + 2 + 3 + 4 + 5

11. Subtraktion? Kein Problem!

Wir schreiben: PRINT 7 – 5

Der Computer schreibt: 2

Wir schreiben: PRINT 25.68 – 37.95

Der Computer schreibt: –12.27

Vervollständigen Sie in den folgenden Zeilen jeweils die Ausgabe des Computers.

Wir schreiben: PRINT 29 – 13

Der Computer schreibt: ....

Wir schreiben PRINT 1500 – 2000

Der Computer schreibt: ....

Wir schreiben: PRINT "1500 – 2000"

Der Computer schreibt: ....

— —

16

–500

1500 – 2000

12. BASIC verwendet + für die Addition und – für die Subtraktion, so wie wir es beim Rechnen mit Papier und Bleistift auch tun. Als Multiplikationszeichen wird in BASIC jedoch der Stern (*) benutzt.

Wir schreiben: PRINT 7 * 5

Der Computer schreibt: 35

Wir schreiben: PRINT 1.23 * 4.567

Der Computer schreibt: 5.61741

Wir schreiben: 9 * 8

Der Computer schreibt: ....

Wir schreiben: PRINT 3.14 * 20

Der Computer schreibt: ....

– –

72
62,8

13. Für die Division wird der Schrägstrich (/) verwendet.

Wir schreiben: PRINT 7/5

Der Computer schreibt: 1.4

Wir schreiben: PRINT 13/16

Der Computer schreibt: 0.8125

Wir schreiben: PRINT 24/3

Der Computer schreibt: ....

Wir schreiben: PRINT 3.14/4

Der Computer schreibt: ....

– –

8
0.785

14. Der Computer gibt das Ergebnis arithmetischer Operationen mit einer Länge von 9 oder 10 Stellen aus. Sofern das tatsächliche Ergebnis eine größere Stellenzahl aufweist, wird die ausgedruckte Zahl auf eine Länge von 9 oder 10 Stellen beschnitten.

Wir schreiben: PRINT 1.2345 * 6.78999

Der Computer schreibt: 8.38224265

Die vollständige Anwort hat 10 Stellen und lautet: 8.382242655

Wir schreiben: PRINT 1/3

Der Computer schreibt: 0.3333333333

Das tatsächliche Ergebnis kann in diesem Fall nicht mit einer endlichen Stellenzahl angegeben werden, sondern ist unendlich lang.

Wir schreiben: PRINT 2/3

Der Computer schreibt: 0.6666666666

Das tatsächliche Resultat wurde nach der zehnten Stelle abgeschnitten.

Nehmen wir an, daß das Ergebnis einer umfangreichen arithmetischen Operation 1.234567898765 lautet. Was wird der Computer ausgeben?
....

Das Ergebnis einer anderen Operation lautet 3.14159265359. Was wird der Computer ausgeben? ....

– –

1.23456789 oder 1.234567898
3.14159265 oder 3.141592653

15. Schreiben Sie PRINT-Statements, um dem Computer den Befehl zu geben, die fogenden numerischen Ausdrücke zu berechnen. Geben Sie auch das vom Computer ausgedruckte Ergebnis an.

Numerischer Ausdruck	PRINT-Statement und Ergebnis
a) 10 + 6	....
b) 15 – 8	....
c) 23 : 5	....
d) 3 x 13	....
e) 8 : 3	....
f) 120 : 7	....

– –

a) PRINT 10 + 6   16
b) PRINT 15 – 8   7
c) PRINT 23/5   4.6
d) PRINT 3*13   39
e) PRINT 8/3   2.66666666 (abgekürzt auf 9 Stellen)
f) PRINT 120/7   17.14295714 (abgekürzt auf zehn Stellen)

16. Der Computer arbeitet arithmetische Ausdrücke von links nach rechts ab, wobei alle Multiplikationen (*) und/oder Divisionen (/) vor Additionen(+) und/oder Subtraktionen (–) ausgeführt werden.

$$\left.\begin{matrix} * \\ / \end{matrix}\right\} \text{ vor } \left\{\begin{matrix} + \\ - \end{matrix}\right.$$

Nachfolgend werden einige BASIC-Ausdrücke angegeben, in denen zwei oder mehr Operationen vorkommen. Für einige dieser Ausdrücke sind bereits die Werte angegeben, die der Computer durch Ausführung der angegebenen Arithmetikoperationen berechnet hat. Vervollständigen Sie bitte den Rest.

Ausdruck	**vom Computer berechnet**	**Rechengang für die arithmetische Operation**
2*3–4	2	2*3–4 = 6 – 4 = 2
2+3*4	14	2+3*4 = 2 + 12 + 14
2*2+4*5	....	....
2+3*4–5	....	....
2*3+4*5–6*7	....	....

– –

26   2*3+4*5 = 6 + 20 = 26
9   2+3*4–5 = 2 + 12 – 5 = 9
– 16   2*3+4*5–6*7 = 6 + 20 – 42 = – 16

17. Hier sind einige weitere Beispiele und Übungen, in denen Division (/) vorkommt. Sehen Sie sich die Beispiele genau an und vervollständigen Sie die Übungen. Folgen Sie dabei der in Abschnitt 16 beschreibenen Reihenfolge, in der der Computer arithmetische Operationen ausführt.

Ausdruck	vom Computer berechnet	Rechengang für die arithmetische Operation
3/4+5	5.75	3/4+5 = .75+5 = 5.75
2–3/4	1.25	2–3/4 = 2–.75 = 1.25
2*3+4/5	6.8	2*3+4/5 = 6+.8 = 6.8
3/4+5*6	....	....
2–2/4+5	....	....

– –

30.75 3/4+5*6 = .75 + 30 = 30.75
6.25 2–3/4+5 = 2–.75 + 5 = 6.25

18. Geben Sie, gemäß den Regeln des Computers, den berechneten Wert für jeden der nachfolgenden Ausdrücke an. (Denken Sie dabei daran, die Berechnung von links nach rechts vorzunehmen!)

Ausdruck	vom Computer berechneter Wert
2*3/4	....
3/4*5	....
3/4/5	....
2*3/4+3/4*5	....

– –

1.5 multiplizieren Sie 2 mit 3 und dividieren Sie das Ergebnis durch 4
3.75 dividieren Sie 3 durch 4 und multiplizieren Sie das Ergebnis mit 5
0.15 dividieren Sie 3 durch 4 und das Ergebnis anschließend durch 5
5.25 berechnen Sie zunächst 2*3/4, anschließend 3/4*5, dann addieren Sie die beiden Teilergebnisse

19. Geben Sie für jeden der folgenden numerischen Ausdrücke die Reihenfolge an, in welcher der Computer die Berechnung ausführt, indem Sie die Ziffern 1, 2 oder 3 in die Kreise über den Operations-Symbolen eintragen. Wir haben die ersten drei bereits für Sie ausgefüllt.

①② a) 2 + 4 – 4 ②① b) 2 – 3 * 4 ①② c) 2 * 3 / 4

○○ d) 2 / 3 / 4

○○○ e) 2 + 3 * 4 – 2

○○○ f) 2 – 3 * 4 + 1

○○○ g) 2 + 3 / 4 * 2

○○ h) 2 + 3 * 4

– –

① ② d) 2 / 3 / 4

② ① ③ e) 2 + 3 * 4 – 2

② ① ③ f) 2 – 3 * 4 + 1

③ ① ② g) 2 + 3 / 4 * 2

② ① h) 2 + 3 * 4

20. Wenn Sie die Reihenfolge ändern möchten, in der die einzelnen Operationen vom Computer ausgeführt werden, müssen Sie Klammern benutzen. Dabei gilt die Regel, daß Operationen in Klammern zuerst ausgeführt werden.

Beginnend mit dem am meisten links liegenden inneren Satz von Klammern führt der Computer, innerhalb zusammgehörender Klammern, die Berechnung wiederum von links nach rechts durch, wobei alle Multiplikationen (*) und/oder Divisionen (/) vor Additionen (+) und/oder Subtraktionen (–) ausgeführt werden.

$$\left.\begin{matrix} * \\ / \end{matrix}\right\} \text{ vor } \left\{\begin{matrix} + \\ - \end{matrix}\right.$$

Sehen Sie sich bei den folgenden Beispielen an, wie die Reihenfolge der einzelnen Operationen völlig verschiedene Ergebnisse liefern kann.

2*3+4 = 10

aber

2*(3+4) = 14 — In diesem Beispiel wird 3+4 zuerst berechnet, da es innerhalb der Klammern liegt und anschließend mit 2 multipliziert wird.

2+3*4+5 = 19

aber

(2+3)*(4+5) = 45 — Berechnen Sie zunächst 2+3, anschließend 4+5 und multiplizieren dann beide Ergebnisse.

Vervollständigen Sie die folgende Übung:

Ausdruck	**vom Computer berechneter Wert**
(2+3)/(4*5)	....
2+3*(4+5)	....
1/(3+5)	....

— —

0.25	(2+3)/(4*5) = 5/20 = .25
29	2+3*(4+5) = 2+3*9 = 2+27 = 29
0.125	1/(3+5) = 1/8 = .125

21. Wir wollen uns jetzt nochmals die Reihenfolge ansehen, in der die Berechnung erfolgt. In den nachstehend angeführten Ausdrücken zeigen die Zahlen in den Kreisen die Reihenfolge an, in der die Operationen ausgeführt werden. Schreiben Sie den endgültigen Wert für jeden Ausdruck hin. Die Berechnung beginnt immer mit dem innersten Satz von Klammern.

Ausdruck — **vom Computer berechneter Wert**

```
 ⑤ ④  ③   ② ①
2 + 3 * (4 – (5 + 6 * 7))     ................................

 ①  ③ ②  ④  ⑤
(3 * 4 + 5 * 6 – 7) / 8       ................................
```

– 127
4.375

22. Ihre nächste Aufgabe besteht darin, ein einwandfreies PRINT-Statement zu schreiben, um dem Computer mitzuteilen, daß er den Wert für jeden der untenstehenden Ausdrücke berechnen und ausdrucken soll. Denken Sie daran, alle Multiplikationen und Divisionen mit den in BASIC gültigen Symbolen anzugeben.

Ausdruck	PRINT-Statement	Ergebnis
2 x 3 + 6 : 7	....	6.85714285
16(33 – 21)	....	192
3.14 x 2 x 2	....	12.56
$\frac{88-52}{18+47}$	....	0.5538461538

```
PRINT 2*3+6/7
PRINT 16*(33–21)
PRINT 3.14*2*2
PRINT (88–52)/(18+47)
```

(Haben Sie auch nicht den Stern für die Multiplikation vergessen?)

23. Geben Sie für jeden der nachfolgenden numerischen Ausdrücke die Reihenfolge der Operationen an, indem Sie 1, 2, 3 usw. in die Kreise oberhalb der jeweiligen Operation eintragen.

```
  ○   ○
a) 2 + (3 – 4)

    ○   ○   ○   ○
b) 2 / (3 / 4) * (2 + 3)

   ○   ○
c) 2 / (3 * 4)

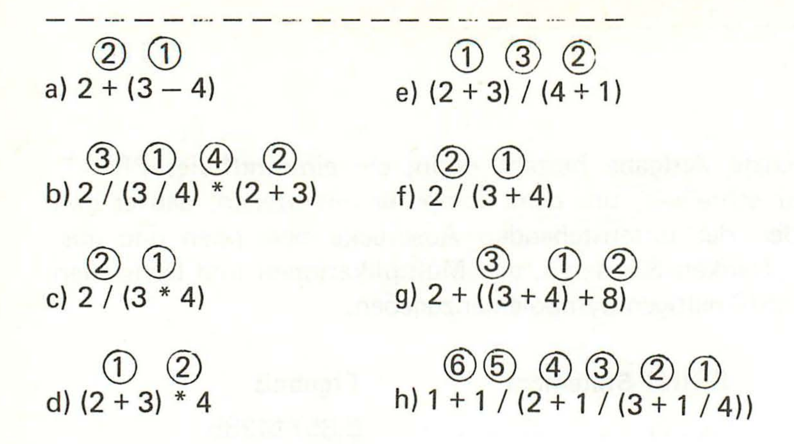
      ○   ○
d) (2 + 3) * 4

      ○   ○   ○
e) (2 + 3) / (4 + 1)

   ○   ○
f) 2 / (3 + 4)

   ○     ○    ○
g) 2 + ((3 + 4) + 8)

   ○   ○    ○   ○    ○   ○
h) 1 + 1 / (2 + 1 / (3 + 1 / 4))
```

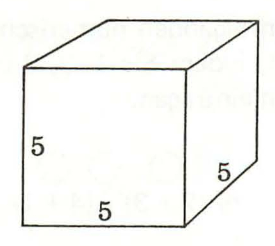


24. Es gibt in BASIC noch ein fünftes arithmetisches Symbol, das die Erhebung einer Zahl in eine Potenz angibt. Diese Operation wird Potenzierung genannt.

^ bedeutet: Erhebe in die nachfolgend angegebene Potenz

SHIFT	→ * ^

Das Volumen eines Würfels ist ein gutes Beispiel dafür, wie Potenzen verwendet werden.



Das Volumen eines Würfels mit der Kantenlänge S ergibt sich zu:

$$V = S^3$$

Ist S = 5 und $V = 5^3$, dann berechnet sich das Volumen aus:

$$V = 5^3 = 5 \times 5 \times 5 = 125$$

Wir drücken nochmals gleichzeitig die Tasten [SHIFT] und [→ * ^] , um dem Computer mitzuteilen, daß er eine Zahl in eine Potenz erheben soll.

Wir schreiben: PRINT 5^3 (5^3 bedeutet 5^3 oder 5x5x5)

Der Computer schreibt: 124.99999998 Hmmmmm . . . . sehen Sie das Ergebnis als 125 an!

Vervollständigen Sie jetzt die folgende Übung und geben Sie Ihre Antworten als ganze Zahlen an.

Wir schreiben: PRINT 2^3

Der Computer schreibt: . . . .

Wir schreiben: PRINT 2^4

Der Computer schreibt: . . . .

Wir schreiben: PRINT 2^5

Der Computer schreibt: . . . .

Wir schreiben: PRINT 7^2

Der Computer schreibt: . . . .

- -

		Der Computer gab an:
8	2 ^ 3 = 2^3 = 2x2x2 = 8	7.99999986
16	2 ^ 4 = 2^4 = 2x2x2x2 = 16	15.9999993
32	2 ^ 5 = 2^5 = 2x2x2x2x2 = 32	31.99993444
49	7 ^ 2 = 7^2 = 7x7 = 49	48.99999631

25. Schreiben Sie ein PRINT-Statement, um jeden der nachfolgend angegebenen Ausdrücke zu berechnen. Wir haben bereits die Ergebnisse geliefert.

Ausdruck	**PRINT-Statement**	**Ergebnis**
3^7	. . . .	2187

8x8x8x8x8	....	32768 (Verwenden Sie ^)
1.06^{10}	....	1.79085
$3^2 + 4^2$	....	24

PRINT 3 ^ 7
PRINT 8 ^ 5 (PRINT 8*8*8*8*8 liefert ebenfalls das gewünschte Ergebnis)

PRINT 1.06 ^ 10

26. Nehmen wir einmal an, wir zahlen 123 DM auf ein Sparbuch ein, für das wir 6% Zinsen pro Jahr, bei jährlicher Berechnung, erhalten. Die Höhe des gesparten Kapitals, einschließlich der Zinsen nach N Jahren, kann mit Hilfe der nachfolgenden Formel berechnet werden.

$$A = P(1 + R/100)^N$$

darin ist:

P = das ursprünglich auf das Sparbuch eingezahlte Grundkapital
R = der jährliche Zinsfuß in Prozent
N = die Spardauer in Jahren
A = die Höhe des Sparkapitals am Ende von N Jahren

In unserem Beispiel ist P = 123 DM, R = 6%. Wir möchten gern die Höhe von A für N = 2, N = 5 und N = 12 Jahre wissen.

Wir schreiben: PRINT 123*(1+6/100) ^ 2
Der Computer schreibt: 130.203 (nach 2 Jahren)

Wir schreiben: PRINT 123*(1+6/100) ^ 5
Der Computer schreibt: 164.602 (nach 5 Jahren)

Jetzt sind Sie an der Reihe. Schreiben Sie das PRINT-Statement für N = 12. Wir haben das vom Computer gelieferte Ergebnis bereits angegeben.

Wir schreiben: ....
Der Computer schreibt: 247.5

Nach 12 Jahren hat sich unser ursprüngliches Kapital also verdoppelt.

```
-----------------------
PRINT 123*(1+6/100)^12
```

27. Computer verwenden zur Darstellung sehr großer oder sehr kleiner Zahlen eine spezielle Schreibweise. Diese Methode zum Schreiben von Zahlen wird "Gleitkomma-Darstellung" genannt. Sie ist in der gleichen Form zu lesen, wie die wissenschaftliche Schreibweise, die gewöhnlich in mathematischen und wissenschaftlichen Lehrbüchern verwendet wird.

Die Bevölkerung der Erde beträgt beispielsweise 40,000,000,000 Menschen:

40 Milliarden = 40,000,000,000

Wir wollen jetzt den Computer auffordern, die Bevölkerung der Erde anzugeben.

Wir schreiben: PRINT 40000000000 (ohne Komma)

Der Computer schreibt: 4E+10

Was haben wir darunter zu verstehen?

Unser ATARI-Computer hat 40 Milliarden als Gleitkomma-Zahl dargestellt, die folgendermaßen zu lesen ist:

4E + 10 = viermal 10 hoch 10, oder
4E + 10 = 4×10^{10}

Die Gleitkomma-Darstellung ist also nichts anderes als eine Kurzschreibweise um sehr große oder sehr kleine Zahlen darzustellen. In der Gleitkomma-Schreibweise wird eine Zahl durch eine "Mantisse" und einen "Exponenten" repräsentiert.

4 = Mantisse
+10 = Exponent

Die Mantisse und der Exponent sind durch den Buchstaben .... getrennt.

E

Anmerkung:

Bei der Zahlen-Eingabe in den Computer dürfen keine Kommas verwendet werden, wie es sonst beim Schreiben von Zahlen üblich ist. Das Komma hat in BASIC PRINT-Statements eine spezielle Bedeutung, die später noch erläutert werden soll.

28. Hier sind einige Beispiele für die Schreibweise von Zahlen in herkömmlicher Art und anschließend in Gleitkommadarstellung.

Eine Trillion:

übliche Schreibweise	1,000,000,000,000
Gleitkommadarstellung	1E+12

Geschwindigkeit einer Schnecke in Meilen/Sekunden:

übliche Schreibweise	0.0000079
Gleitkommadarstellung	7.9E–6 (negativer Exponent)

Unterstreichen Sie in den beiden obigen Gleitkommazahlen die Mantisse und kreisen Sie den Exponenten ein.

---

1 E(+ 12)
7.9 E(–6)

29. Hier sind einige weitere Beispiele, die zeigen, wie der ATARI-Computer Gleitkommazahlen ausdruckt.

Wir schreiben:	PRINT 123456789
Der Computer schreibt:	123456789
Wir schreiben:	PRINT 12345678899
Der Computer schreibt:	1.2345678E+10
Wir schreiben:	PRINT 12345678000
Der Computer schreibt:	1.2345678E+10

Unser ATARI Computer druckt wenigstens neun Stellen für die Mantisse und schneidet sie an der neunten Stelle ab. Die Mantisse wird als eine Zahl mit einer von Null verschiedenen Ziffern links vom Dezimalpunkt und einer Länge von bis zu 8 Stellen rechts vom Dezimalpunkt ausgegeben.

Wir schreiben: PRINT 333333333333

Der Computer schreibt: ....

Wir schreiben: PRINT 666666666666

Der Computer schreibt: ....

3.33333333E+11
6.66666666E+11

30. In gleicher Weise werden sehr kleine Zahlen dargestellt.

Wir schreiben: PRINT .0000001234567889

Der Computer schreibt: 1.23456788E–07 (negativer Exponent)

Wir schreiben: PRINT .0000001234567884

Der Computer schreibt: 1.23456788E–07 (negativer Exponent)

Wieviele Stellen hat die Mantisse links vom Dezimalpunkt? ....
Wieviele Stellen hat sie maximal rechts vom Dezimalpunkt? ....

1
8

(Beispiele für Mantissen mit weniger als 8 Stellen rechts vom Dezimalpunkt finden Sie in den abschnitten 28 und 29).

31. Zahlen in Gleitkommadarstellung können in die übliche Schreibweise umgeformt werden. Wenn der Exponent positiv ist, gehen Sie folgendermaßen vor:

a) Schreiben Sie die Mantisse separat.

b) Verschieben Sie den Dezimalpunkt der Mantisse um soviele Stellen zum rechten Ende der Mantisse, wie es der Exponent angibt. Falls nötig hängen Sie Nullen an.

Beispiel: 6.12345678E+08

a) 6.12345678 b) 6.123456789. (Verschieben Sie den Dezimalpunkt um 8 Stellen)

Somit erhält man: 6.12345678E+08 = 612,345,678

Beispiel: 4E+09

a) 4. b) 4.000000000. (verschieben Sie den Dezimalpunkt um 9 Stellen und fügen Sie 9 Nullen hinzu).

Damit erhält man: 4E+09 = 4,000,000,000

Versuchen Sie es jetzt einmal selbst: 1,2345678E+13

a) .... b) ....

Daher ist 1.2345678E+13 = ...

---

a) 1.2345678 b) 1.2345678000000 (Verschieben Sie den Dezimalpunkt um 13 Stellen).

12,345,678,000,000

32. Ist der Exponent negativ, brauchen wir nur die Richtung zu ändern, in der wir den Dezimalpunkt verschieben.

a) Schreiben Sie die Mantisse separat.

b) Verschieben Sie den Dezimalpunkt der Mantisse um soviele Stellen nach links, wie es der Exponent angibt. Falls nötig, füllen Sie leere Stellen mit Nullen auf.

Beispiel: 7.9E–06

a) 7.9 b) .000007.9 (Verschieben Sie den Dezimalpunkt um 6 Stellen und fügen Sie fünf Nullen hinzu).

Daher ist 7.9E–06 = .0000079

Jetzt sind Sie an der Reihe: 1.23456E–08

a) .... b) ....

Daher ist 1.23456E–08 = ...

---

a) 1.23456
b) .00000001.23456
.00000001.23456

33. Schreiben Sie die nachfolgenden Gleitkommazahlen in üblicher Darstellung.

Gleitkomma	**übliche Schreibweise**
a) 1.23456E+06	....
b) 1.23456E–06	....
c) 1.23456E+07	....
d) 1.23456E–09	....
e) 1E+11	....
f) 1E–11	....

---

a) 1234560 oder 1,234,560
b) .00000123456
c) 12345600 oder 12,345,600
d) .00000000123456
e) 100,000,000,000
f) .00000000001

34. Schreiben Sie die nachfolgenden Zahlen in Gleitkommadarstellung um.

Übliche Schreibweise	**Gleitkommadarstellung**
a) 1,234,560,000	....
b) .000000123456	....
c) 10,000,000,000	....
d) .000000001	....
e) 1234567888888	....

f) .00000123456788888 ....
g) 6.02 x 10^{21} ....
h) 1.67 x 10^{-11} ....

Anmerkung:

Die Zahlen unter (g) und (h) sind in wissenschaftlicher Schreibweise angegeben.

\- -

a) 1.23456E+09
b) 1.23456E–07
c) 1E+10
d) 1E–09
e) 1.23456788E+12 (Mantisse auf 9 Stellen abgeschnitten)
f) 1.23456788E–06 (Mantisse auf 9 Stellen abgeschnitten)
g) 6.02E+21
h) 1.67E–11

35. Vielleicht haben Sie schon einmal von der alten Geschichte gehört, in der ein weiser Mann einem reichen König einen großen Dienst erwiesen hat. Der König fragte ihn daraufhin, welche Belohnung ihm denn angemessen erscheinen würde. Die Bitte des Weisen war sehr einfach. Er bat nur um Weiszenkörner, deren Anzahl folgendermaßen berechnet werden sollte: für das erste Feld eines Schachbretts ein Korn, für das zweite Feld zwei Körner, für das dritte Feld 4 Körner Weizen usw., so daß mit jedem Feld eine Verdoppelung der Weizenmenge erfolgt. Die Menge wächst daher gemäß nachfolgender Aufstellung an:

Feld-Nummer	Anzahl der Weizenkörner
1	1
2	2
3	4
5	16
6	32
7	64

Und so weiter. Für das Feld N ergibt sich damit eine Anzahl von 2^{N-1} Körnern. Wir wollen jetzt einmal ermitteln, wieviele Körner es auf dem

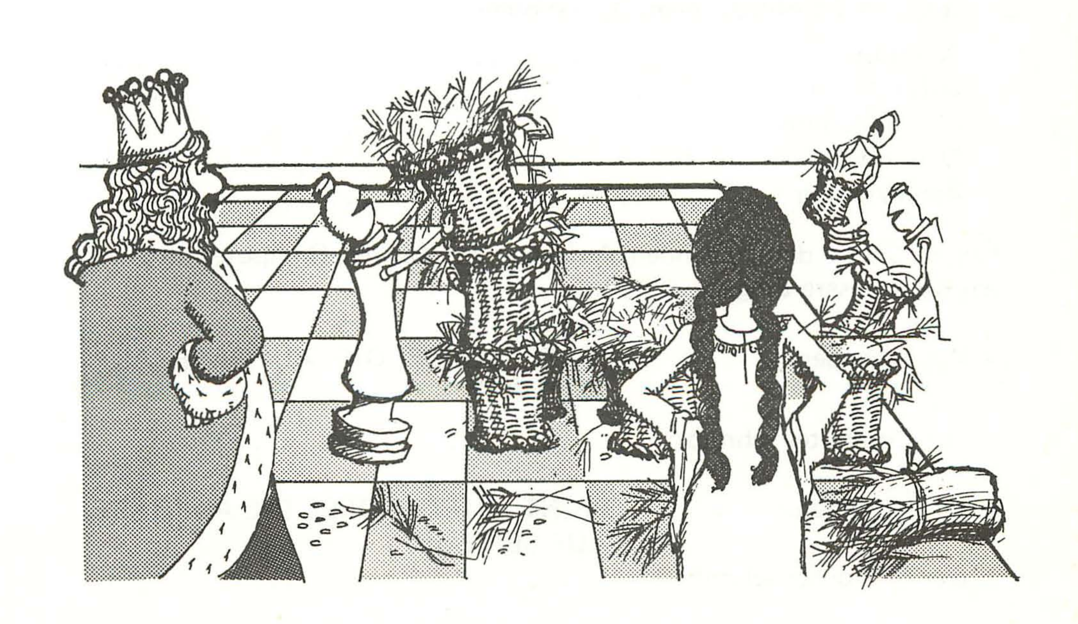
sechzehnten Feld sind.

Wir schreiben:	PRINT 2 ^ 15 (Da N = 16 ist N–1 = 15)
Der Computer schreibt:	32768

Jetzt sind Sie an der Reihe. Schreiben Sie ein PRINT-Statement, um zu ermitteln, wieviele Weizenkörner für das Feld mit der Nummer 64 benötigt werden.

Wir schreiben:	....
Der Computer schreibt:	9.2233704E+18 (Das ist eine Menge Weizen!)

PRINT 2 ^ 63



EINGENTEST

Arbeiten Sie nun den nachfolgenden Eigentest durch, um festzustellen, wieviel Sie bis jetzt gelernt haben.

1. In diesem Kapitel haben wir direkte Statements bentzt. Direkte Statements haben keine Zeilennummern. Was macht der Computer, wenn wir ein direktes Statement eintippen und auf RETURN drücken? ....
   ....

2. Nehmen wir an, Sie geben ein Statement in den Computer ein und machen einen Tippfehler. Wie würden Sie den Fehler korrigieren? ...
   ....
   ....

3. Schreiben Sie die in ATARI BASIC verwendeten Symbole für die folgenden arithmetischen Operationen:

   Addition ....
   Subtraktion ....
   Multiplikation ....
   Division ....
   Potenzierung ....

Geben Sie in den folgenden Aufgaben an, was der Computer als Antwort auf unsere Eingabe schreibt.

4. Wir schreiben: BITTE LOESE DIE AUFGABE AUF SEITE 157

   Der Computer schreibt: ....

5. Wir schreiben: PRINT "BITTE LOESE DIE AUFGABE AUF SEITE 157"

   Der Computer schreibt: ....

6. Wir schreiben: PRINT 9*37/5+32
   Der Computer schreibt: ....

7. Nachfolgend ist die Formel zur Umwandlung von Temperaturangaben in Fahrenheit in Celsiusgrade angegeben.

$$C = \frac{5}{9}(F - 32$$

(C = Grad Celsius, F = Grad Fahrenheit)

Schreiben Sie ein PRINT-Statement, durch das der Computer veranlaßt wird, die Temperatur in Grad Celsius für 72 Grad Fahrenheit zu berechnen und auszudrucken. ....

....

8. Geben Sie an, was der Computer schreibt.

Wir schreiben: PRINY "THE QUICK RED FOX JU

[SHIFT] [DELETE BACK S]

Der Computer schreibt: ....

9. Welchen Befehl gibt das nachfolgende Statement dem Computer? PRINT 23 + 45 ....

....

10. Geben Sie an, welche der nachfolgenden mathematischen Ausdrücke in BASIC-Schreibweise korrekt sind.

a) 2*3*4*5 ....
b) 7(8 + 9) ....
c) 1.23 : 4.567 ....
d) 3^2 + 4^2 ....
e) $\frac{1}{37 - 29}$ ....
f) $(1 + 7/100)^3$ ....
g) 2^2^2 ....

11. Schreiben Sie für jeden unkorrekt geschriebenen numerischen Ausdruck in Aufgabe 10 einen gültigen BASIC-Ausdruck. ....

....

12. Geben Sie für jeden der nachfolgenden numerischen Ausdrücke die Reihenfolge an, in welcher der Computer die Berechnung vornimmt, indem Sie die Zahlen 1, 2, 3 usw. in die Kreise über den

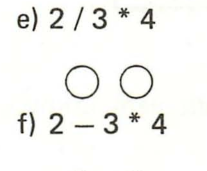
Operations-Symbolen eintragen.

○ ○
a) 2 + 4 − 4

○ ○
b) 2 * 3 * 4

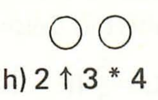
○ ○
c) 2 * 3 / 4

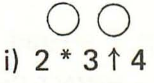
○ ○
d) 2 / 3 / 4

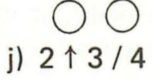

○ ○
e) 2 / 3 * 4

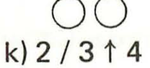
○ ○
f) 2 − 3 * 4

○ ○
g) 2 + 3 / 4


○ ○
h) 2 ↑ 3 * 4


○ ○
i) 2 * 3 ↑ 4


○ ○
j) 2 ↑ 3 / 4


○ ○
k) 2 / 3 ↑ 4

○ ○
l) 2 ↑ 3 ↑ 4

○ ○ ○
m) 1 + 2 * 3 ↑ 4

n) 2 * 3 − 4 ↑ 3 + 5 ↑ 2

13. Geben Sie für jeden der nachfolgenden numerischen Ausdrücke die Reihenfolge der Operationen an, indem Sie die Zahlen 1, 2, 3 usw. in die Kreise oberhalb der Operationssymbole eintragen.

○ ○
a) 2 + (3 − 4)

○ ○
b) 2 / (3 /4)

○ ○
c) 2 / (3 * 4)

○ ○
d) (2 − 3) * 4

○ ○
e) (2 + 3) / 4

○ ○
f) 2 ↑ (3 * 4)

○ ○
g) 2 ↑ (3 ↑ 4)

○ ○ ○ ○ ○ ○
h) 1 + 1 / (2 + 1 / (3 + 1 / 4))

14. Schreiben Sie jede der nachfolgenden Gleitkommazahlen in der üblichen Schreibweise.

Gleitkommadarstellung	übliche Schreibweise
a) 1.23456E+05	....
b) 1.23456E−05	....
c) 1.23456E+08	....
d) 1.23456E−08	....
e) 1E+12	....
f) 1E−12	....

15. Schreiben Sie die nachfolgenden, in üblicher mathematischer Schreibweise angegebenen Zahlen, in ATARI-BASIC Gleitkommadarstellung.

übliche Schreibweise	Gleitkommadarstellung
a) 1,234,560,000,000	....
b) .00000000123456	....
c) 10,000,000,000	....
d) .0000000001	....
e) 12345678999	....
f) .00000123456789	....
g) 6.02×10^{23}	....
h) 1.67×10^{-21}	....

Anmerkung:

Die Zahlen (g) und (h) sind in wissenschaftlicher Schreibweise angegeben.

Antworten zum Eigentest

Die jeweils am Schluß der Antwort in Klammern stehenden Nummern beziehen sich auf die jeweiligen Abschnitte, in denen der abgefragte Stoff behandelt wurde.

1. Der Computer führt das Statement aus und "vergißt" es dann (5).

2. Drücken Sie die Taste DELETE BACK S um den Fehler zu beseitigen und tippen Sie dann die korrekten Zeichen ein (7 – 9).

3. Addition +
   Subtraktion –
   Multiplikation *
   Division /
   Potenzierung ∧ (Abschnitte 10 – 24)

4. ERROR– BITTE LOESE DIE AUFGABE (Der Computer versteht nicht, was wir von ihm wollen.) (3)

5. BITTE LOESE DIE AUFGABE AUF SEITE 157 (5 und 6)

6. 98.6 (Abschnitte 16 – 24)

7. PRINT 5/9*(F–32) oder PRINT (5/9)*(F–32). Haben Sie sich an den Stern erinnert? Übrigens wäre auch die Antwort PRINT 5*(F–32)/9 richtig gewesen. (Abschnitte 20 – 23)

8. Nur der Cursor ist zu sehen. Die ganze Zeile ist gelöscht (Abschnitt 9).

9. Berechne den numerischen Ausdruck 23+45 und gebe das Ergebnis aus. Der Computer addiert 23 und 45, was 68 ergibt und zeigt (druckt) auf dem Bildschirm 68 (Abschnitt 10) an.

10. Die Ausdrücke (a), (d) und (g) sind gültig. In (g) berechnet der Computer den Ausdruck 2∧ 2∧ 2 folgendermaßen:

    2∧2∧2 = 4∧2 = 16

    Mit anderen Worten, der Computer arbeitet den Ausdruck von links nach rechts ab, als wenn er lauten würde: (2 ∧ 2) ∧ 2 (Abschnitte 24 – 26).

11. b) 7*(8+9)
    c) 1.23/4.567
    e) 1/(37–29)
    f) (1+7/100)∧3 (Abschnitte 16 – 26)

12.

① ②
a) 2 + 4 – 4

① ②
b) 2 * 3 * 4

① ②
c) 2 * 3 / 4

① ②
d) 2 / 3 / 4

① ②
e) 2 / 3 * 4

② ①
f) 2 – 3 * 4

② ①
g) 2 + 3 / 4

① ②
h) 2 ↑ 3 * 4

② ①
i) 2 * 3 ↑ 4

① ②
j) 2 ↑ 3 / 4

② ①
k) 2 / 3 ↑ 4

① ②
l) 2 ↑ 3 ↑ 4

③ ② ①
m) 1 + 2 * 3 ↑ 4

③ ④ ① ⑤ ②
n) 2 * 3 – 4 ↑ 3 + 5 ↑ 2

(Abschnitte 16 – 25)

13.

② ①
a) 2 + (3 – 4)

② ①
b) 2 / (3 / 4)

② ①
c) 2 / (3 * 4)

① ②
d) (2 – 3) * 4

① ②
e) (2 + 3) / 4

② ①
f) 2 ↑ (3 * 4)

② ①
g) 2 ↑ (3 ↑ 4)

⑥ ⑤ ④ ③ ② ①
h) 1 + 1 / (2 + 1 / (3 + 1 / 4))

(Abschnitte 16 – 25)

14. a) 123,456
   b) .0000123456
   c) 123,456,000
   d) .0000000123456
   e) 1,000,000,000,000
   f) .000000000001 (Abschnitte 29 – 34)

15. a) 1.23456E+12
   b) 1.23456E–09
   c) 1E+10
   d) 1E–10
   e) 1.23456789E+10
   f) 1.23456789E–06
   g) 6.02E+23
   h) 1.67E–21 (Abschnitte 29 – 34)

Kapitel 3

Zuordnungs-Statements, gespeicherte Programme, Verzweigung

Dieses Kapitel soll Sie mit einigen der wichtigsten BASIC-Statements vertraut machen. Von jetzt an können wir mit interessanteren Programmen arbeiten, um die vielseitigen Anwendungsmöglichkeiten Ihres ATARI-Computers zu demonstrieren.

In diesem Kapitel werden Sie die Funktionen und das Format der nachfolgend angeführten Statements und Kommandos kennenlernen:

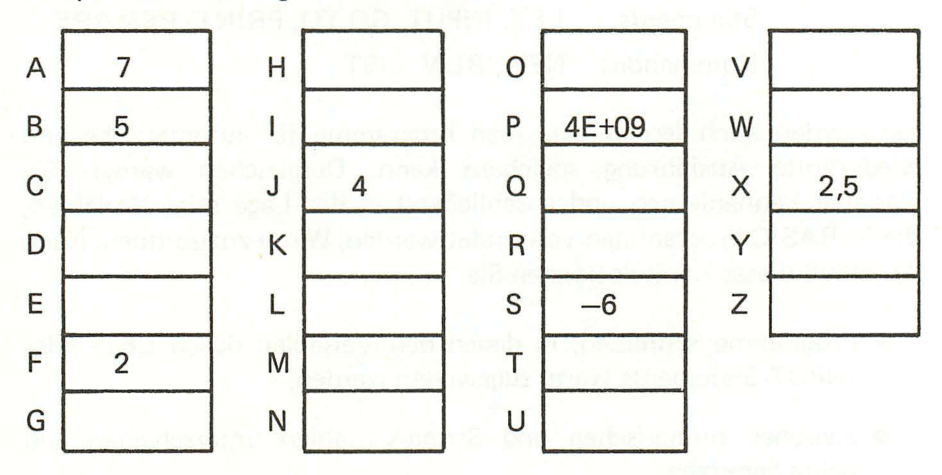
Statements	LET, INPUT, GO TO, PRINT, REMARK
Kommandos	NEW, RUN, LIST

Sie werden auch lernen, wie man Programme für automatische und wiederholte Ausführung speichern kann. Desgleichen werden Sie Variable kennenlernen und anschließend in der Lage sein, Variablen, die in BASIC-Programmen verwendet werden, Werte zuzuordnen. Nach Abschluß dieses Kapitels können Sie ....

- Programme schreiben, in denen den Variablen durch LET- oder INPUT-Statements Werte zugewiesen werden;
- zwischen numerischen und String-Variablen unterscheiden und beide benutzen;
- Programme schreiben, in denen ein Wert durch einen BASIC-Ausdruck berechnet und einer numerischen Variablen in einem LET-Statement zugeordnet wird;
- ein BASIC-Programm im Speicher des Computers speichern;
- ein unerwünschtes Programm aus dem Computer-Speicher löschen, ein zur Zeit im Computer befindliches Programm auflisten und ein Programm ausführen lassen (RUN);
- Statements in einem gespeicherten Programm editieren, korrigieren und löschen;

- Programme schreiben, die GO TO Statements benutzen, um einen Teil eines Programms zu wiederholen oder ein Programmstück zu überspringen.
- REMARK-Statements benutzen, um Programme für den Menschen leichter lesbar und verständlicher zu machen;
- PRINT-Statements schreiben, die ausgedruckte Ergebnisse durch zusätzliche erklärende Mitteilungen kenntlich machen;

1. Um das Prinzip der Variablen und die Funktion des LET-Statements in BASIC zu verdeutlichen, stellen Sie sich vor, daß innerhalb des Computers 26 kleine Kästchen vorgesehen sind. Jedes kann zu jedem Zeitpunkt eine einzige Zahl enthalten.



Wir haben in einigen Kästchen bereits Zahlen gespeichert. Beispielsweise befindet sich eine 7 in der Schachtel A und eine 5 in der Schachtel B.

a) Welche Zahl befindet sich in Schachtel F? ....
b) In J? ....
c) –6 befindet sich in Schachtel ....
d) 2.5 befindet sich in Schachtel ....
e) Welches Kästchen enthält eine Gleitkommazahl? ....

– –

a) 2 b) 4 c) S d) X
e) P (Die Gleitkommazahl lautet 4E+09).

2. Die Kästchen C und N sind anschließend einzeln dargestellt. Benutzen Sie einen Bleistift, um folgende Aufgaben durchzuführen:

a) "Legen" Sie eine 8 in Schachtel C. Mit anderen Worten, schreiben Sie die Ziffer "8" in die mit "C" bezeichnete Schachtel. Führen Sie diesen Schritt aus, bevor Sie zu (b) weitergehen.

b) "Legen" Sie 12 in N. Führen Sie diesen Schritt aus, bevor Sie zu (c) weitergehen.

c) "Legen" Sie 27 in N. Halt! Eine Schachtel kann zu einem bestimmten Zeitpunkt nur eine Zahl enthalten. Bevor Sie 27 in N eingeben können, müssen Sie daher zuerst die 12 löschen, die Sie zuvor eingegeben haben.

C [ ]　　　　N [ ]

Füllen Sie die Kästchen aus, bevor Sie sich die Antwort ansehen.

– –

C [8]　　　　N [27]

3. Wenn der Computer eine Zahl in eine der Schachteln hineinlegt, löscht er automatisch den bisherigen Inhalt dieser Schachteln, genau so wie Sie es getan haben. Um 27 in N zu legen, haben Sie zuerst den bisherigen Inhalt 12 ausradiert.

Wir nennen A, B, C, ... Z Variable. Die Zahl in der Schachtel A ist der Wert von A, ebenso die Zahl in Schachtel B der Wert von B usw.

Um den Computer anzuweisen, "eine Zahl in eine der Schachteln zu legen", verwenden wir das LET-Statement. Um es etwas technischer auszudrücken sagen wir, daß wir einer Variablen einen numerischen Wert zuordnen.

Wir schreiben: LET A = 7 ("Lege" 7 in Schachtel A.)
Wir schreiben: PRINT A (Drucke den Inhalt von Schachtel A)
Der Computer schreibt: 7

In diesem Programm ist .... die Variable und .... der Wert, der ihr durch das LET-Statement zugeordnet wurde.

A; 7

4. Vervollständigen Sie bitte folgende Zeilen:

a) Wir schreiben: LET X = 23
PRINT X

Der Computer schreibt: ....

b) Wir schreiben: LET Z = –1
PRINT Z

Der Computer schreibt: ....

c) Wir schreiben: LET A = 1
LET A = 2
PRINT A

Der Computer schreibt: ....

d) Wir schreiben: LET D = 7
LET W = D
PRINT W

Der Computer schreibt: ....

---

a) 23
b) –1
c) 2 (Das Statement LET A = 2 ersetzt den zuvor durch LET A=1 zugewiesenen Wert.)
d) 7 (LET W = D überträgt den Wert von D in W. Der Weg befindet sich jedoch auch immer noch in D.)

5. Schreiben Sie ein LET-Statement, um P den Wert 3.14 zuzuordnen sowie ein PRINT-Statement, um den Wert von P auszudrucken.

Wir schreiben: ....
....
Der Computer schreibt: ....

```
LET P = 3.14
PRINT P
3.14
```

Anmerkung:

In ATARI BASIC kann das Wort LET auch weggelassen werden. Beispielsweise können wir auch P=3.14 an Stelle von LET P=3.14 schreiben.

6. Sobald wir einer Variablen einen Wert zugewiesen haben, können wir sie in mathematischen Ausdrücken verwenden.

Wir schreiben: LET A = 7
LET B = 5

Wir haben jetzt eine 7 in "Schachtel" A und eine 5 in "Schachtel" B.

Wir schreiben: PRINT A + B
Der Computer schreibt: 12 (Da A = 7 u. B = 5 ist A+B = 7+5 = 12)

Wir schreiben: PRINT A – B
Der Computer schreibt: 2 (Da A = 7 u. B = 5, ist A–B = 7–5 =2)

Jetzt vervollständigen Sie die folgenden Zeilen:

Wir schreiben: PRINT A*B
Der Computer schreibt: ....

Wir schreiben: PRINT A/B
Der Computer schreibt: ....

35
1.4

7. Bei den Variablen, die wir bis jetzt benutzt haben, handelt es sich um die sogenannte numerische Variable. Der einer numerischen Variablen zugeordnete Wert muß eine Zahl sein.

BASIC kennt aber noch eine andere Variablen-Type, die als String-Varialbe bezeichnet wird. Der einer String-Variablen zugeordnete Wert muß ein String sein, also eine Folge von Zeichen. Eine String-Variable ist durch einen Buchstaben gekennzeichnet, dem ein Dollar-Zeichen ($) angehängt ist.

Bevor wir String-Variable benutzen, müssen wir dem Computer zunächst mitteilen, wie lang der String sein wird. Das heißt, wir müssen ihm sagen, wieviel Speicherplatz er zur Aufnahme des Strings bereitstellen muß. Wir benutzen dazu das DIM-Statement (Dimension-Statement). Beispielsweise weist das folgende DIM-Statement den Computer an, Speicherplatz für einen String mit einer Länge von bis zu 5 Zeichen bereitzustellen.

Wir schreiben:	DIM N$(5)
Der Computer schreibt:	READY

Wir haben dem Computer damit mitgeteilt, daß eine String-Variable mit der Bezeichnung N$ eine Länge von bis zu 5 Zeichen haben kann.

Wir schreiben: LET N$ = "JERRY"

Damit wird der String-Variablen N$ der Wert JERRY zugeordnet.

N$ | JERRY |

Jetzt wollen wir den Wert von N$ ausdrucken.

Wir schreiben:	PRINT N$
Der Computer schreibt:	JERRY

Vervollständigen Sie jetzt bitte folgende Zeilen:

Wir schreiben:

```
DIM Z$(100)
LET Z$ = "MEIN BENUTZER
VERSTEHT MICH"
PRINT Z$
```

Der Computer schreibt: ....

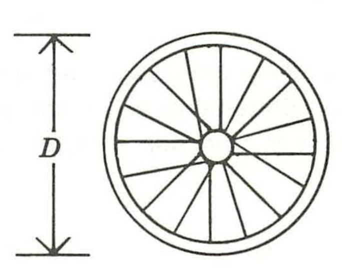
MEIN BENUTZER VERSTEHT MICH

Unser DIM-Statement ließ Platz für bis zu 100 Zeichen reservieren. Der tatsächliche String ist in diesem Beispiel allerdings kürzer als 100 Zeichen. Wesentlaich ausführlicher werden Strings in Kapitel 9 beschrieben.

8. Jetzt wollen wir uns mit einem anderen Problem befassen. Wir haben drei Fahrräder mit Raddurchmessern von 16–, 24– und 26 Zoll. Von jedem Rad möchten wir wissen, welchen Weg es bei einer Umdrehung zurücklegt. Diese Strecke ist natürlich der Umfang des Rades. Wir wollen im Folgenden D für den Durchmesser und C für den Umfang des Rades verwenden.

$C = \pi D$

wobei $\pi = 3.14159....$ ist.



Wir werden uns mit 3.14 als einer groben Annäherung für π begnügen. Das beschriebene Problem läßt sich auf folgende Weise lösen:

Wir schreiben: PRINT 3.14*16
Der Computer: 50.24

Wir schreiben: PRINT 3.14*24
Der Computer: 75.36

Wir schreiben: PRINT 3.14*26
Der Computer: 81.64

Es gibt aber noch eine weitere Möglichkeit, um diese Berechnung auszuführen. Füllen Sie dazu in den folgenden Zeilen die freien Stellen aus:

Wir schreiben: LET D = 16
PRINT 3.14*D
Der Computer: 50.24

Wir schreiben: LET D = 24
PRINT 3.14*D
Der Computer: ....

Wir schreiben: LET ....
....

Der Computer: 81.64

———————————————————

75.36
D = 26
PRINT 3.14*D

9. Jetzt sind wir in der Lage, den großen Schrittt von einzelnen, direkt ausgeführten Statements zu einem gespeicherten Programm zu machen. Wir wollen das folgende Programm im Speicher unseres Computers ablegen.

```
10 LET D = 16
20 PRINT 3.14 * D
```

Dieses Programm besteht aus zwei Statements, jedes in einer einzelnen Zeile. Jedes Statement beginnt mit einer Zeilennummer. In unserem Beispiel sind es die Zahlen 10 und 20. Eine Zeilennummer kann eine beliebige ganze Zahl zwischen 1 und 32767 sein.

Wenn wir Statements mit Zeilennummern schreiben, werden sie beim Drücken von RETURN zunächst nicht ausgeführt. Stattdessen werden die Statements im Speicher des Computers für spätere Ausführung untergebracht.

Wie Sie im Kapitel 2 gelernt haben, werden Statements ohne Zeilennummern als direkte Statements bezeichnet. Der Computer führt ein direktes Statement sofort aus und "vergißt" es anschließend. Dies geschieht jedoch nicht, wenn wir ein Statement mit einer Zeilennummer schreiben. Was geschieht statt dessen? ....
....

———————————————————

Das Statement wird für spätere Ausführung im Speicher des Computers abgelegt. Das heißt, der Computer "erinnert" sich dann an das State-

ment.

10. Die Zeilennummern geben dem Computer die Reihenfolge an, in der er den Statements im Programm folgen soll. Dabei ist es nicht erforderlich, daß die Zeilennummern aufeinanderfolgende ganze Zahlen wie 1, 2, 3, 4, 5 usw. sind. Es ist vielmehr üblich, in Zehnerschritten zu numerieren, wie wir es auch im folgenden Programm tun. Wir können dann, sofern wir es wünschen, später leicht ein oder mehr Statements in das Programm zwischen bereits bestehende Statements einfügen.

```
10 LET D = 16
20 PRINT 3.14 * D
```

Wie viele zusätzliche Zeilen könnten zwischen Zeile 10 und Zeile 20 eingefügt werden? . . . .

— —

9 (Zeilen 11, 12, 13, 14, 15, 16, 17, 18, 19)

Natürlich müssen Sie die Numerierung nicht in Zehnerschritten vornehmen. Sollten Sie Abstände von 13, oder 5 oder sonst einer Zahl vorziehen, dann tun Sie das ruhig. Es ändert prinzipiell nichts.

11. Bevor wir ein Programm speichern, müssen wir zunächst alle alten Programme entfernen oder löschen, die sich evtl. bereits im Speicher des Computers befinden.

Wir schreiben: NEW (Und drücken natürlich auf RETURN !)
Der Computer: READY

Der Computer hat damit den Bereich eines Speichers gelöscht, der BASIC-Programme aufnimmt. Er ist jetzt zum Speichern neuer Programme bereit.

Wie löschen oder entfernen wir ein altes Programm aus dem Computer-Speicher? . . . .

— —

Wir schreiben NEW und drücken auf die RETURN-Taste.

Sollten wir uns bei der Eingabe von NEW verschreiben, erhalten wir

eine Fehler-Meldung, beispielsweise:

Wir schreiben: GNU
Der Computer: ERROR– GNU ■

Unser Computer mag offensichtlich keine Gnus, aber wenn wir jetzt NEW richtig eintippen und auf RETURN drücken, ist wieder alles in Ordnung.

12. Jetzt sind wir in der Lage, unseren Programm-Zweizeiler aus den Abschnitten 9 und 10 zu speichern.

Zuerst müssen wir alle alten Programme löschen, indem wir .... schreiben und die Taste RETURN betätigen. Um das Programm zu speichern, schreiben wir zunächst die erste Zeile, bzw. das erste Statement und betätigen dann die RETURN-Taste. Anschließend schreiben wir die zweite Zeile und ....

NEW; betätigen die RETURN-Taste

13. Versuchen wir das einmal.

Wir schreiben: NEW
Der Computer: READY
Wir schreiben: 10 LET D = 16
20 PRINT 3.14 * D

Wenn wir einen Tippfehler machen, beseitigen wir ihn in der schon gewohnten Weise. Das Programm ist jetzt gespeichert, und der Computer wartet geduldig auf unser nächstes Statement oder Kommando.

Hmmm . . . . , ob das Programm wohl wirklich im Computer gespeichert ist? Wir können das herausfinden, indem wir LIST eintippen und die Taste RETURN betätigen.

Wir schreiben: LIST und betätigen RETURN
Der Computer: 10 LET D = 16
20 PRINT 3.14 * D
READY

Das Kommando LIST fordert den Computer also auf, alle gegenwärtig gespeicherten Statements auszudrucken. Nach dem Auflisten des Programms meldet er READY, um uns zu sagen, daß wir wieder an der Reihe sind. Wenn kein Programm gespeichert ist, schreibt der Computer nur READY.

Wie können wir den Computer dazu auffordern, das gegenwärtig in seinem Speicher befindliche Programm auszudrucken? . . . .
. . . .

— —

Wir schreiben LIST und drücken die Taste RETURN

14. Das Auflisten eines BASIC-Programmes zeigt uns, ob sich ein Programm im Speicher des Computers befindet oder nicht. Vielleicht wollen wir weitere Statements zu dem Programm hinzufügen, oder uns ansehen, ob wir die Statements korrekt geschrieben haben.

Wenn wir möchten, daß der Computer das Programm ausführt, schreiben wir . . . . Wünschen wir, daß der Computer mitteilt, ob in seinem Speicher ein Programm untergebracht ist, und er das Programm ausdrucken soll, dann schreiben wir . . . .

— —

RUN; LIST

15. Nachdem Sie RUN eingegeben haben, um den Computer aufzufordern, das Programm auszuführen, bzw. nachdem Sie LIST eingetippt haben, um dem Computer zu befehlen, uns mitzuteilen, was sich in seinem Speicher befindet, müssen Sie noch etwas tun, damit der Computer antwortet, und zwar? . . . .

— —

Die Taste RETURN drücken.

16. Nehmen wir einmal an, Sie haben unser kleines Programm in den Computer eingegeben. Jetzt möchten Sie es ausführen. Schreiben Sie RUN und drücken Sie auf RETURN.

Wir schreiben: RUN und drücken auf RETURN.

Der Computer: 50.24
READY

Der Computer hat das Programm damit ausgeführt und die einzelnen Statements in der Reihenfolge ihrer Zeilennummern abgearbeitet. Zuerst wurde der Computer dazu aufgefordert, einer Variablen mit dem Statement 10 LET D=16 einen Wert zuzuordnen. Das bedeutet, daß der Computer den Wert .... in die "Schachtel" gelegt hat, die durch die Variable .... gekennzeichnet ist.

– –

16; D

17. Nachdem er die Anweisungen im Statement mit der Zeilennummer 10 ausgeführt hat, wendet sich der Computer dem Statement mit der nächst höheren Zeilennummer zu: 20 PRINT 3.14*D. Ein Teil dieser Zeile fordert den Computer auf, zwei Zahlen miteinander zu multiplizieren. Eine davon ist direkt in dem Statement angegeben. Die andere Zahl ist an einem Platz gespeichert, der durch die Variable D gekennzeichnet ist. Um welche beiden Zahlen handelt es sich, die der Computer multiplizieren soll? ....

– –

3.14 und 16, da 16 der Wert von D ist.

18. Im letzten Abschnitt haben wir gesehen, daß ein Teil von Zeile 20 den Computer auffordert, zwei Zahlen miteinander zu mulitplizieren. Welchen Befehl gibt Zeile 20 dem Computer außerdem? ......
....

– –

Das Ergebnis der Multiplikation auszudrucken.

Anmerkung:

Einige Computer benötigen am Ende eines Programms ein END-Statement, wie es nachfolgend gezeigt wird. Der ATARI-Computer benötigt END nicht, Sie können es jedoch nach Belieben vorsehen.

```
10 LET D = 16
20 PRINT 3.14 * D
30 END
```

19. Wir wollen uns jetzt noch einmal ansehen, was alles geschieht, wenn wir unser kleines Programm zur Berechnung des Umfangs eines Fahrradreifens benutzen.

NEW	Zuerst löschen wir alle alten Programme
READY	
10 LET D = 16 20 PRINT 3.14 * D	Dann schreiben wir dieses Programm, das aus zwei Statements besteht.
LIST	Als nächstes fordern wir den Computer auf, das Programm aufzulisten.
10 LET D = 16 20 PRINT 3.14 * D	Der Computer listet das Programm auf (druckt es aus).
READY	Dann schreibt er READY und hält an.
RUN	Schließlich fordern wir den Computer auf, das Programm abzuarbeiten.
50.24	Er führt das Programm aus,
READY	schreibt dann wieder READY auf den Bildschirm und hält an.

Das Programm ist jetzt immer noch im Speicher des Computers vorhanden. Was wird geschehen, wenn wir erneut RUN eintippen? ....

....

— —

Der Computer führt das Programm nochmals aus. Da im Programm nichts geändert wurde, gibt der Computer das gleiche Ergebnis wie zuvor aus (50.24).

20. Was geschieht, wenn wir bei der Eingabe des Programms einen Fehler machen?

Wir schreiben:	10 LET F = 16 20 PRINT 3.14 * D	Wir haben also F anstatt D geschrieben.

Wir schreiben: RUN

Der Computer: 0

Hoppla! Da ist etwas falsch gelaufen. Sehen wir uns doch noch einmal das Programm an.

Wir schreiben: LIST

Der Computer:
```
10 LET F = 16
20 PRINT 3.14 * D
```
Da ist es ganz deutlich zu sehen: wir haben in Zeile 10 versehentlich auf F statt D getippt.

Wir schreiben: 10 LET D = 16

Wir schreiben Zeile 10 jetzt nochmals richtig. Sie ersetzt die alte Zeile 10.

Wir schreiben: LIST

Der Computer:
```
10 LET D = 16
20 PRINT 3.14 * D
```
Dadurch vergewissern wir uns, daß die alte Zeile 10 tatsächlich durch die neue Zeile ersetzt wurde. Dabei sehen wir auch gleichzeitig, ob unsere Eingabe diesmal richtig war.

Nehmen wir einmal an, wir hätten den Fehler sofort nach dem Schreiben erkannt. Wie hätten wir ihn korrigieren können? ....
....

Wir hätten DELETE BACK S drücken und dann den richtigen Buchstaben eingeben können.

21. Wir wollen einmal annehmen, unser Programm zur Umfangsberechnung sei immer noch im Speicher vorhanden. jetzt möchten wir aber gern den Wert ändern, der D zugewiesen ist. Dazu ersetzen wir Zeile 10 durch eine neue Zeile 10. Nachdem wir diese Änderung vorgenommen haben, listen wir das Programm auf.

Wir schreiben:
```
10 LET D = 24
LIST
```

Der Computer:
```
10 LET D = 24
20 PRINT 3.14 * D
```
Hier sehen Sie die neue Zeile 10, sowie die unverändert gebliebene alte Zeile 20.

Wie ersetzen wir eine Zeile in einem gespeicherten Programm? ....
....

— —

Wir schreiben eine neue Zeile mit der Nummer derjenigen Zeile, die wir ersetzen wollen.

Anmerkung:

Wir geben nicht NEW ein, da wir möchten, daß Zeile 20 im Speicher stehen bleibt, während wir nur Zeile 10 ändern. NEW würde das Löschen beider Statements im Computer-Speicher bewirken.

22. Wir wollen jetzt einmal das modifizierte Programm ablaufen lassen.

Wir schreiben:	RUN
Der Computer:	75.36

> Das Wort READY soll von jetzt ab in diesem Buch weggelassen werden.

Jetzt sind Sie an der Reihe.

Wir schreiben: 10 LET D = 26
LIST
Der Computer: ....
....
Wir schreiben: RUN
Der Computer: ....

— —

```
10 LET D = 26            (neue Zeile 10)
20 PRINT 3.14 * D        (alte Zeile 20)
81.64
```

23. Wie schnell wächst Ihr Sparguthaben an?
Wir haben folgendes Problem vorliegen: Wir zahlen P DM auf ein Sparkonto ein, für das wir R% Zinsen im Jahr erhalten, die jährlich berechnet werden. Wieviel Geld haben wir nach dem Ablauf von N Jahren auf unserem Sparbuch?

Hier sind noch einmal alle verwendeten Größen zusammengestellt:

P Anfangskapital

R jährlicher Zinsfuß in %
N die Anzahl der Jahre
A Sparkapital und Zinsen nach N Jahren

Die Berechnung von A erfolgt mit der nachstehend angegebenen Formel:

$$A = P(1 + R/100)^N$$

Mit dieser Beziehung kann A, bei gegebenem P, R und N ermittelt werden.

Anschließend sehen Sie ein Programm zur Berechnung von A für P = 1000 DM, R = 6% und N = 3 Jahre. Vervollständigen Sie Zeile 40 in korrekter BASIC-Schreibweise, unter Verwendung der oben angegebenen Formel.

```
10 LET P = 1000
20 LET R = 6
30 LET N = 3
40 LET A = ..........................................
50 PRINT A
```

———————————————————

```
P*(1+R/100)^N
```

(Haben Sie sich an den Stern und das Zeichen ∧ für die Potenzierung erinnert?)

24. Wir wollen das Programm jetzt speichern und ablaufen lassen. Anschließend sollen Sie einige Änderungen vornehmen.

Wir schreiben: NEW

Anmerkung:

Bitte denken Sie daran, daß wir in Zukunft das an dieser Stelle vom Computer angegebene READY weglassen wollen. Außerdem ist allmählich wohl auch der weitere Hinweis auf das Betätigen der Taste RETURN nach der Eingabe von Statements oder Kommandos wie RUN, LIST oder NEW überflüssig.

```
10 LET P = 1000
```

```
20 LET R = 6
30 LET N = 3
40 LET A = P*(1+R/100) ∧ N
50 PRINT A
RUN
```

Der Computer: 1191.01596

Geben Sie jetzt an, wie Zeile 30 für N=5 geändert werden muß.

Wir schreiben: ....

RUN

Der Computer: 1338.22554 Wert von A für P=1000, R=6, N=5

– –

30 LET N = 5

25. Ändern Sie jetzt das Programm so, daß R=7% und N=8 Jahre ist.

Wir schreiben: ....

....

RUN

Der Computer: 1718.19

– –

20 LET R=7
30 LET N=8

26. Schreiben Sie jetzt einmal auf, was der Computer ausgibt, wenn wir LIST eintippen.

Wir schreiben: LIST

Der Computer: ....

....

....

....

....

– –

10 LET P=1000
20 LET R=7

```
30 LET N=8
40 LET A=P*(1+R/100) ^ N
50 PRINT A
```

27. LET-Statements sind ja gut und schön, aber wie umständlich ist es doch, jedesmal alle Zeilen neu eintippen zu müssen, wenn man die Werte der Variablen ändern möchte. Zum Glück bietet BASIC uns mit dem INPUT-Statement eine sehr gute Lösung für dieses Problem.

Das INPUT-Statement ermöglicht es dem Computer-Benutzer, bei jedem Lauf des Programms den Variablen unterschiedliche Werte zuzuweisen, ohne daß das Programm selbst modifiziert werden müßte. Wenn der Comuter im Programm auf ein INPUT-Statement trifft, schreibt er ein Fragezeichen auf den Bildschirm und wartet darauf, daß der Benutzer für eine Variable einen Wert eingibt. Nachfolgend finden Sie ein Beispiel dafür.

Wir schreiben:	NEW 10 INPUT A 20 PRINT A RUN	Zuerst löschen wir alle alten Programme, und geben das neue ein. Der Computer wird durch RUN zur Ausführung aufgefordert.
Der Computer:	? ■	Er gibt ein Fragezeichen aus, schaltet den Cursor ein und wartet dann.

Was hat der Computer gemacht, als wir RUN eingetippt haben? ......

....

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Er druckte ein Fragezeichen und wartete darauf, daß wir etwas tun.

Anmerkung:

Das Fragezeichen ist eine Möglichkeit eine Aufforderung an den Benutzer zu richten. Der Compuer gibt eine derartige Aufforderung aus, um uns mitzuteilen, daß wir an der Reihe sind, etwas zu tun. Die Meldung READY ist ein anderes Beispiel für eine Aufforderung.

28. Das INPUT-Statement veranlaßt den Computer, ein Fragezeichen zu drucken, dann anzuhalten und zu warten. Er wartet auf einen Wert,

den er der Variablen zuweisen kann, die im INPUT-Statement erscheint. Computer sind sehr geduItig. Wenn wir nicht mit ihm zusammenarbeiten, indem wir einen Wert eintippen, dann wartet er, und wartet, und wartet . . . .

Arbeiten wir daher mit unserem stets geduldigen Computer zusammen und geben einen Wert für A ein. Wir wollen jetzt 3 für den Wert eintippen, der A zugeordent werden soll und drücken anschließend auf RETURN. Der Computer legt dann den von uns eingegebenen Wert in "Schachtel" A und fährt mit dem Programm fort.

```
NEW
10 INPUT A
20 PRINT A
RUN
? 3
 3
```

Nachdem wir 3 als Wert für A eingegeben und dann auf RETURN gedrückt hatten, ging der Computer weiter zu Zeile 20 und druckte unseren Wert aus.

Nach der Eingabe von RUN und Betätigen der RETURN-Taste druckte der Computer ein . . . . Wir haben dann 3 eingetippt, den Wert für unsere Variable . . . . . Der Computer führte anschließend das PRINT A Statement (Zeile 20) aus und druckte den . . . . von A.

– –

Fragezeichen; A; Wert

29. Das Programm kann man anschließend nochmals mit einem neuen Wert für A laufen lassen. Tun Sie einmal so, als wären Sie der Computer und geben Sie an, wie ein Programm-Lauf aussehen würde, wenn Ihr menschlicher Computer-Benutzer 23 als Wert von A eingibt.

```
RUN
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

– –

```
?23
 23
```

30. Wir wollen jetzt ein INPUT-Statement dazu verwenden, um die Daten für das uns schon vertraute Fahrrad-Problem einzugeben. Mit Daten werden im allgemeinen Informationen bezeichnet, die von einem Computer-Programm benutzt werden.

Wir schreiben:

```
NEW
10 INPUT D
20 PRINT 3.14*D
RUN
```

Der Computer:

```
?
```

Der Computer möchte von uns einen Wert für den Durchmesser D des Rades. Anschließend wird er die Strecke berechnen, die bei einer Umdrehung des Rades zurückgelegt wird und sie ausdrucken. Wir wollen diese Berechnung für Durchmesser von D=16, D=24 und D=26 durchführen.

Zuerst lassen wir das Programm für D=16 ablaufen.

```
RUN
?16
 50.24
```

Als der Computer ein Fragezeichen druckte, haben wir 16 als Wert von D eingegeben. Der Computer berechnete und druckte 3.14*D und hielt dann an.

Jetzt tippen wir noch einmal RUN ein. Wenn der Computer wieder ein Fragezeichen ausgibt, tippen wir anschließend 24 als Wert für D ein.

```
RUN
?24
 75.36
```

Der Comuter führt die Berechnung aus, zeigt das Ergebnis auf dem Bildschirm an und stoppt.

Jetzt sind Sie an der Reihe. Vervollständigen Sie den dritten Programm-

Lauf.

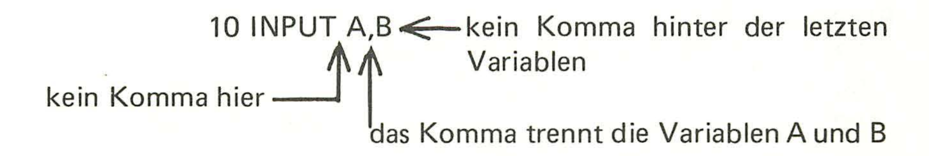
```
            RUN
            .......................................
            .......................................
```

```
?26
 81.64
```

31. Ein IPUT-Statement kann auch dazu verwendet werden, zwei oder mehr Variablen Werte zuzuweisen.

```
10 INPUT A,B      (zwei Variable A und B)
20 PRINT A+B
RUN
? 7,5             (zwei Werte, 7 und 5)
 12
```

Als der Computer ein Fragezeichen druckte, haben wir zwei durch ein Komma getrennte Zahlen eingetippt und dann die Taste RETURN betätigt. Der Computer wies die erste Zahl der Variablen A als Wert, die zweite der Variablen B zu. Beachten Sie bitte folgende, wichtigen Hinweise:



RUN
?7,5 ← kein Komma nach der letzten Zahl
↑
Komma zwischen den beiden Zahlen

Der Wert 7 wird der Variablen A, der Wert 5 der Variablen B zugeordnet.

Bitte füllen Sie in der folgenden Zusammenstellung die leeren Stellen aus:

Wenn ein Programm abläuft, das ein INPUT-Statement enthält, wird die nach Erscheinen des Fragezeichens zuerst eingegebene Zahl der .... Variablen im INPUT-Statement zugeordnet; der ...... vom Benutzer eingetippte Wert wird der zweiten Variablen im INPUT-Statement zugewiesen usw. Beide Variablen im INPUT-Statement des Programms und die vom Benutzer beim Lauf des Programms eingetippten Werte müssen durch .... getrennt werden.

\- -

ersten; zweite; Kommas

32. Hier sehen Sie einen weiteren RUN des Programms in Abschnitt 31. Jetzt wollen wir jedoch die Zahlen 73 als Wert von A und 59 als Wert von B eingeben.

RUN
? 73 — Zu dumm! Jetzt haben wir versehentlich zu früh die Taste RETURN getrückt!
? ■

Der Computer gibt ein weiteres Fragezeichen aus und schaltet den Cursor ein. Das soll bedeuten: "Haben Sie nicht etwas vergessen?" Wir können den Programm-Lauf dann durch Eingabe der zweiten Zahl, dem Wert von B vervollständigen.

```
RUN
? 73
? 59
 132
```

Was geschieht, wenn wir nicht für jede Variable in einem INPUT-Statement einen numerischen Wert eintippen? ....

\- -

Der Computer gibt ein weiteres Fragezeichen aus.

Anmerkung:

Wenn Sie mehr Zahlen eingeben, als Variable vorhanden sind, ignoriert der Computer die restlichen Werte.

33. Nehmen wir einmal an, Sie stehen mit Ihren Freunden, lauter begeisterten Fahrradfahrern, um Ihren Computer herum und alle bewundern Ihre neuen Programmierkünste. Sie beschließen, Ihren Freunden zu demonstrieren, wie der Computer arbeitet. Dazu wollen Sie das Programm zur Berechnung des Umfanges eines Fahrradreifens benutzen. Das umständliche dabei ist jedoch, daß Sie für jeden Ihrer Freunde einen eigenen Programmlauf durchführen müssen. Daher halt! Fügen Sie zu Ihrem Programm erst ein neues Statement hinzu.

```
10 INPUT D
20 PRINT 3.14*D
30 GO TO 10
```

Das ist etwas Neues - das GO TO-Statement.

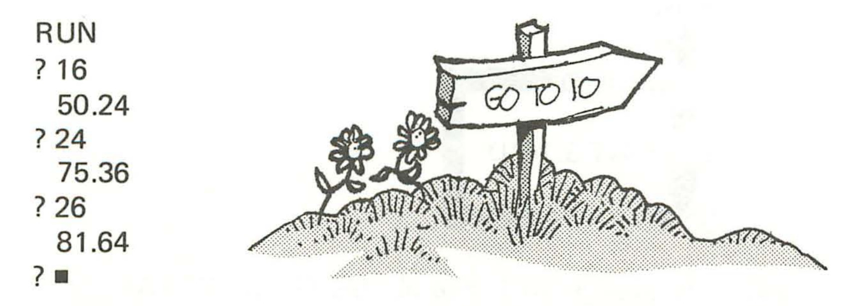
In BASIC fordert ein GO TO-Statement den Computer auf, zu der Zeilennummer, die den Worten GO TO folgt, zu springen bzw. dorthin zu verzweigen und dann in der durch die Zeilennummern vorgegebenen Reihenfolge die weiteren Instruktionen abzuarbeiten.

Im obigen Programm fordert das GO TO-Statement den Computer auf, zur Zeile . . . . zu springen und erneut mit dem Programm zu beginnen.

10

34. Jetzt wollen wir einmal sehen, was geschieht, wenn wir das Programm laufen lassen. (Wir nehmen dazu an, daß Sie NEW geschrieben und dann das Programm eingetippt haben.)

```
RUN
? 16
 50.24
? 24
 75.36
? 26
 81.64
? ■
```



Nach jedem Fragezeichen haben wir einen Wert für D eingetippt. Der Computer berechnete anschließend 3.14*D, druckte das Ergebnis und gab eine weiteres Fragezeichen aus.

Damit haben wir jetzt alle gewünschten Berechnungen ausgeführt, aber der Computer weiß nicht, daß wir bereits fertig sind. Er steht in der Programmzeile 10 und wartet auf INPUT-Daten. Wie können wir diesen Zustand beenden?

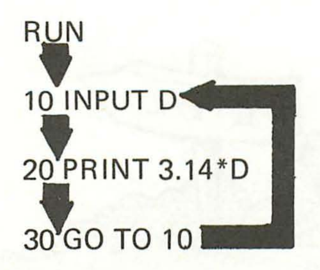
Ganz einfach! Drücken Sie auf die Taste **BREAK**

?	wir betätigen die Taste BREAK
STOPPED AT LINE 10	Der Computer gibt die Zeile an, in der
■	BREAK erfolgte und schaltet den Cursor ein.

Nehmen wir an, der Computer führt ein INPUT-Statement aus und hat ein Fragezeichen angezeigt. Was macht er, wenn wir die Taste BREAK drücken? ....

Der Computer tritt aus dem INPUT-Statement aus, schreibt STOPPED AT n (wobei n die Zeilennummer angibt, an der BREAK erfolgte) und schaltet den Cursor ein.

35. Erinnern Sie sich: BASIC Statements werden in der Reihenfolge der Zeilennummern ausgeführt, bis ein GO TO-Statement die Reihenfolge ändert. In dem Programm aus Abschnitt 33 werden die Statements in der nachfolgend durch die Pfeile gekennzeichneten Reihenfolge ausgeführt.



Dieser Vorgang läuft solange im Kreise ab, bis Sie die BREAK-Taste betätigen.

Das obige Programm ist ein Beispiel für eine Schleife. Das GO TO-Statement veranlaßt den Computer, zum Anfang zurückzuspringen. In

unserem speziellen Beispiel geht er zu einem INPUT-Statement und hält dort an. Wir können aber auch Programme mit "Non-Stop-Schleifen" schreiben, wie das folgende Beispiel zeigt. Geben Sie erst NEW ein, bevor Sie das folgende Programm eintippen.

RUN

13 PRINT "UM MICH ANZUHALTEN, DRUECKE DIE TASTE 'BREAK' "

27 GO TO 13

Lassen Sie das Programm noch nicht laufen! Zeichnen Sie statt dessen Pfeile ein, wie wir es im vorigen Beispiel getan haben, um zu verdeutlichen, wie der Computer das Programm ausführt.

RUN

13 PRINT "UM MICH ANZUHALTEN, DRUECKE DIE
TASTE 'BREAK' "

27 GO TO 13

Wir haben in diesem Beispiel bewußt die Zeilennummern 13 und 27 verwendet, um Sie daran zu erinnern, daß Sie nicht in Schritten von 10 durchnumerieren müssen, obwohl wir das meist so machen.

36. Jetzt wollen wir das Programm von Abschnitt 35 eingeben und laufen lassen. Vergessen Sie nicht die Taste BREAK zu drücken, um den Computer anzuhalten.

```
13 PRINT "UM MICH ANZUHALTEN, DRUECKE DIE TASTE 'BREAK' "
27 GO TO 13

RUN
UM MICH ANZUHALTEN, DRUECKE DIE TASTE 'BREAK'
UM MICH ANZUHALTEN, DRUECKE DIE TASTE 'BREAK'
UM MICH ANZUHALTEN, DRUECKE DIE TASTE 'BREAK'
```

STOPPED AT LINE 13 (ODER 27)

Jetzt haben wir die Taste BREAK betätigt.

Der Computer führte die Zeile 13 immer wieder aus, da ihn das GO TO-Statement in Zeile 27 ständig dazu aufforderte, zur Zeile 13 zurückzugehen. Diese Schleife ist damit sozusagen eine "ewige" Schleife, da sie immer wieder durchlaufen wird, bis sie jemand unterbricht. Manchmal wird eine derartige Schleife auch als "unendliche" Schleife bezeichnet, da sie nicht automatisch anhält.

Wie können wir den Computer unterbrechen oder anhalten, wenn er ein Programm ausführt? . . . .

– –

Wir betätigen die Taste BREAK.

37. Nachdem wir jetzt wissen, wie wir einen "wildgewordenen" Computer durch Betätigen der Taste BREAK anhalten können, wollen wir ein lustiges Computer-Programm schreiben. Dieses Programm soll den Bildschirm schnell mit "Gelächter" anfüllen. Denken Sie daran, vor der Eingabe des neuen Programms das alte mit NEW zu löschen. Dann geben Sie das folgende Programm ein. Es enthält einige bereits bekannte Statements, wie PRINT und GO TO, sowie ein neues mit REMARK bezeichnetes Statement. Wir benutzen das Statement, um dem Menschen, der das Programm lesen will, etwas über das Programm mitzuteilen. Der Computer selbst ignoriert die REMARK-Statements einfach.

```
100 REMARK FUELLE DEN BILDSCHIRM MIT 'GELAECHTER'
110 PRINT "HA HA";
120 GO TO 110
```

Haben Sie auch nicht das Semikolon am Ende von Zeile 110 vergessen? Nein? Gut, dann lassen Sie das Programm jetzt durch Eingabe von RUN ablaufen. Hat sich der Bildschirm wirklich mit "Gelächter" gefüllt? . . . .

– –

Wir denken, Sie haben mit Ja geantwortet. Als wir das Programm ausprobiert haben, füllte sich der gesamte Bildschirm sehr schnell mit

"HA HA".

38. Versuchen Sie das obige Programm noch einmal; modifizieren Sie aber zuvor die Zeile 110 in folgender Weise:

110 PRINT "HA HA"

Diesmal entfällt das Semikolon. Lassen Sie das Programm dann erneut ablaufen. Hat sich diesmal nur der linke Rand des Bildschirms mit HA HA gefüllt? ....

Wir sind sicher, Sie haben wiederum mit Ja geantwortet. Das Semikolon veranlaßt den Computer, quer über den gesamten Bildschirm eine ganze Zeile von links nach rechts auszufüllen. Sobald der Computer am rechten Rand angelangt ist, fährt er einfach in der nächsten Zeile fort. Ohne das Semikolon beginnt der Computer jedoch nach jedem PRINT Statement mit einer neuen Zeile.

39. Das REMARK-Statement hat auch eine Kurzform, die Sie ebenfalls verwenden können. Die Kurzform besteht einfach aus den ersten drei Buchstaben von REMARK, lautet also REM. Bei der Ausführung eines Programms beachtet der Computer nur die drei ersten Buchstaben. Sofern es sich dabei um REM handelt, springt er weiter zum folgenden Statement mit der nächsthöheren Zeilenzahl, ohne den Rest des mit REM begonnenen Statements weiter zu beachten. Daher würde es auch die gleiche Wirkung haben, wenn Sie REMEMBER anstatt REMark schreiben würden. Schreiben Sie Zeile 100 in Abschnitt 37 um und verwenden Sie dabei die Kurzform des REMark-Statements.

....

----

100 REM FUELLE DEN SCHIRM MIT 'GELAECHTER'

40. Wenn der Computer ein PRINT-Statement ausführt, das nur die Zeilennummer, gefolgt von dem Wort PRINT enthält, hat dieses Statement die gleiche Wirkung, als wenn Sie die Taste RETURN drücken würden. Auf dem Bildschirm wird dadurch eine Leerzeile eingefügt. Ein "leeres" PRINT-Statement ist daher sehr zweckmäßig, um ein

Programm leichter lesbar zu machen.

Hier sehen Sie nochmals einen Lauf unseres Fahrradprogramms aus den Abschnitten 33 und 34.

```
RUN

?16
 50.24
?24
 75.36
? ■
```

Das sieht alles ein bißchen zusammengequetscht aus, finden Sie nicht auch?

Versuchen wir noch einmal, die Ausgabe etwas besser lesbar zu machen. Dazu fügen wir in das Programm das folgende "leere" PRINT-Statement ein.

```
25 PRINT
```

Wir schreiben:

```
10 INPUT D
20 PRINT 3.14*D
25 PRINT
30 GO TO 10
```

Lassen wir das Programm jetzt noch einmal ablaufen und sehen uns an, was geschieht.

```
RUN

?16
 50.24

?24
 75.36

? ■
```

Sind Ihnen die Zwischenräume durch die Leerzeilen aufgefallen? Sie

wurden durch das von uns zusätzlich eingefügte PRINT-Statement verursacht.

Vergleichen Sie einmal beide Läufe des Programms. Welche Anweisung gibt die Zeile 25 PRINT dem Computer? ....

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Füge eine Leerzeile auf dem Bildschirm, nach dem Ausdrucken des Wertes von 3.14*D ein. Das geschieht, bevor der Computer GO TO 10 druckt.

41. Da wir jetzt schon einmal dabei sind, die Ausgabe lesbarer zu gestalten, wollen wir versuchen, unser mit INPUT-Statements arbeitendes Programm noch lesbarer zu machen. Wenn wir ein INPUT-Statement in einem Programm vorsehen, wäre es sehr vorteilhaft, den Benutzer irgendwie darauf hinzuweisen, wonach das INPUT-Statement fragt. Wir wollen dazu unserem Fahrrad-Programm das folgende Statement hinzufügen:

```
5 PRINT "RADDURCHMESSER" ;
```

Sehen Sie das Semikolon am Ende des PRINT-Statements? Wenn am Ende eines PRINT-Statements ein Semikolon vorgesehen ist, bleibt der Computer in der gleichen Reihe des Bildschirms, anstatt am Anfang der nächsten Zeile fortzufahren. Hier sehen Sie unser bearbeitetes Programm:

```
 5 PRINT "RADDURCHMESSER" ;
10 INPUT D
20 PRINT 3.14*D
25 PRINT
30 GO TO 10
```

Lassen Sie jetzt das modifizierte Programm ablaufen.

Wir schreiben: RUN
Der Computer: RADDURCHMESSER?

(das Wort stammt aus Zeile 5)

(das Fragezeichen wird von dem INPUT-Statement in Zeile 10 bewirkt)

Jetzt sehen wir endlich genau, was der Computer von uns wissen möchte. Er will von uns einen Wert für den RADDURCHMESSER. Nun gut, geben wir also 16 ein und drücken RETURN. Schreiben Sie bitte auf, was als nächstes geschieht:

```
RADDURCHMESSER? 16
.................................
.................................
------------------------
50.24
RADDURCHMESSER? ■
```

42. Was der Computer möchte, ist jetzt eindeutig durch den in Anführungszeichen stehenden String im PRINT-Statement gesagt. In gleicher Weise wollen wir die vom Computer ausgeführte Berechung kennzeichenen (siehe Zeile 20).

```
 5 PRINT "RADDURCHMESSER";
10 INPUT D
20 PRINT "RADUMFANG" ; 3.14*D
25 PRINT
30 GO TO 10
```

Das Statement 20 PRINT "RADUMFANG"; 3.14*D fordert den Computer auf:

1) den String RADUMFANG auszudrucken und dann
2) den Wert von 3.14*D zu berechnen und auszugeben.

Versuchen wir das einmal. Füllen Sie dazu die leeren Zeilen im folgenden Programmablauf aus:

```
RUN

RADDURCHMESSER? 16
RADUMFANG 50.24

RADDURCHMESSER? 24
RADUMFANG ......................
```

RADDURCHMESSER? 26

. . . .

. . . .

— —

75.36
RADUMFANG 81.54

RADDURCHMESSER? ■

43. Schreiben Sie das Programm in Abschnitt 42 in geeigneter Weise so um, damit ein Lauf so aussieht, wie nachfolgend angegeben.

RUN
WENN SIE DEN DURCHMESSER EINES RADES IHRES FAHRRADS EINGEBEN, SAGE ICH IHNEN, WELCHE STRECKE SIE BEI EINER UMDREHUNG DES RADES ZURUECKLEGEN.

RADDURCHMESSER? 16
DIE BEI EINER UMDREHUNG ZURUECKGELEGTE STRECKE BETRAEGT 50.24

RADDURCHMESSER? 24
DIE BEI EINER UMDREHUNG ZURUECKGELEGTE STRECKE BETRAEGT 75.36

RADDURCHMESSER?

. . . . und so weiter. Um das Programm anzuhalten, drücken Sie auf die Taste BREAK.

— —

```
10 REMARK FAHRRAD PROGRAMM
20 PRINT "WENN SIE DEN DURCHMESSER EINES RADES IHRES
         FAHRRADS"
30 PRINT "EINGEBEN, SAGE ICH IHNEN, WELCHE STRECKE SIE
         BEI"
40 PRINT "EINER UMDREHUNG DES RADES ZURUECKLEGEN"
50 PRINT
60 PRINT "RADDURCHMESSER";
```

```
70 INPUT D
80 PRINT "DIE BEI EINER UMDREHUNG ZURUECKGELEGTE
        STRECKE BETRAEGT"; 3.14*D
90 GOTO 50
```

44. Das INPUT-Statement gehört, genau wie das LET-Statement, zur Gruppe der als Zuweisungs-Statements bezeichneten BASIC-Instruktionen. Der Wert, den wir als Antwort auf ein Fragezeichen eintippen, wird der Variablen im INPUT-Statement zugeordnet. Sofern es sich bei der INPUT-Variablen um eine String-Variable handelt, können wir einen String als Wert eingeben. Er wird dann der String-Variablen zugeordnet. Dafür finden Sie nachstehend ein einfaches Beispiel. Bitte füllen Sie die Stellen aus, die wir leer gelassen haben.

```
 5 DIM N$(100)
10 PRINT "WIE HEISSEN SIE?";
20 INPUT N$
30 PRINT N$
40 PRINT
50 GO TO 10
RUN
WIE HEISSEN SIE? JERALD R. BROWN
JERALD R. BROWN
WIE HEISSEN SIE? LEROY FINKEL
 .............................
WIE HEISSEN SIE? FIREDRAKE, DER DRACHEN
 .............................
```

LEROY FINKEL
FIREDRAKE, DER DRACHEN

In Kapitel 9 werden die Strings wesentlich ausführlicher behandelt.

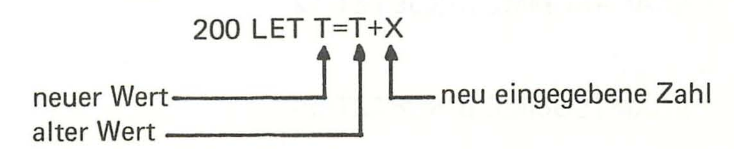
45. Hier lernen Sie ein Programm kennen, mit dem Sie den Computer als Addiermaschine benutzen können, und zwar durch wiederholte Ausführung einer "Addierroutine" mit Hilfe einer GO TO-Schleife. Unter einer "Routine" versteht man eine Folge mehrerer Statements, die eine bestimmte arithmetische Operation ausführen.

```
100 REM DIE TEUERSTE ADDIERMASCHINE DER WELT
110 PRINT "ICH BIN DIE TEUERSTE ADDIERMASCHINE DER
          WELT."
120 PRINT "JEDES MAL, WENN ICH 'X=?' SCHREIBE, GEBEN"
130 PRINT "SIE EINE ZAHL EIN UND BETAETIGEN RETURN."
140 PRINT "ICH WERDE DANN DIE SUMME ALLER BISHER VON"
150 PRINT "IHNEN EINGEGEBENEN ZAHLEN AUSRUCKEN."
160 LET T=0
170 PRINT
180 PRINT "X=";
190 INPUT X
200 LET T=T+X
210 PRINT "GESAMTSUMME BIS JETZT";T
220 GO TO 170
```

Die Zeilen 170 bis 220 bilden eine GO TO-Schleife. Sie werden für jede vom Benutzer eingegebene Zahl ausgeführt.

Beachten Sie die LET-Statements in den Zeilen 160 und 200, in denen jeweils die Variable T vorkommt. Zeile 160 liegt außerhalb der GO TO-Schleife. Sie wird einmal ausgeführt, bevor die Schleife beginnt, wodurch T auf 0 gesetzt wird. Dieser Vorgang wird Initialisierung genannt. Durch die Initialisierung erhält eine Variable einen Anfangswert.

Zeile 200 liegt innerhalb der GO TO-Schleife. Daher wird sie bei jedem Schleifendurchlauf ausgeführt. In Zeile 200 wird ein neuer Wert für T errechnet, indem der alte, in der "Schachtel" T abgelegte Wert, zu der vom Computer-Benutzer als Wert für die INPUT-Variable X eingegebenen Zahl addiert wird.



a) Nehmen Sie an, der alte Wert von T ist 0 und die eingegebene Zahl X ist 12.
Wie groß ist der neue Wert von T? . . . .

b) Nehmen Sie an, der alte Wert von T ist 12 und für X wird 43 einge-

geben.
Wie groß ist der neue Wert von T? ....

- -

a) 12 b) 55

46. Achten Sie darauf, wie die PRINT-Statements (Zeilen 110 – 150) dazu verwendet werden, dem Benutzer eine Erklärung zu geben und instruktionen für das Programm zu liefern. Liegen diese PRINT-Statements innerhalb oder außerhalb der Schleife? ....

- -

außerhalb

Die Zeilen 110 – 150 werden daher nur einmal ausgeführt und zwar dann, wenn Sie zum erstenmal RUN eingeben. Wenn Sie natürlich durch BREAK den Computer unterbrechen und das Programm nochmals starten, werden die Zeilen 110 – 150 erneut ausgeführt.

47. Jetzt wollen wir das Programm einmal ablaufen lassen und sehen, wie es arbeitet.

```
RUN
ICH BIN DIE TEUERSTE ADDIERMASCHINE DER WELT
JEDES MAL, WENN ICH X=? SCHREIBE, GEBEN
SIE EINE ZAHL EIN UND BETAETIGEN RETURN.
ICH WERDE DANN DIE SUMME ALLER BISHER VON
IHNEN EINGEGEBENEN ZAHLEN AUSDRUCKEN.
X=? 12
GESAMTSUMME BIS JETZT 12

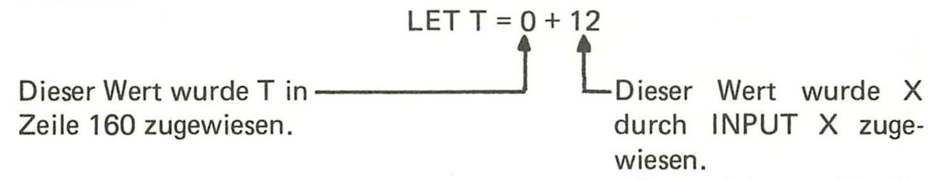
X=? 43
GESAMTSUMME BIS JETZT 55

X=? ■
```

Erinnern Sie sich noch daran, wie Sie den Computer daran hindern können, eine neue Schleife zu durchlaufen? Wenn nicht, sehen Sie sich Abschnitt 34 noch einmal an.

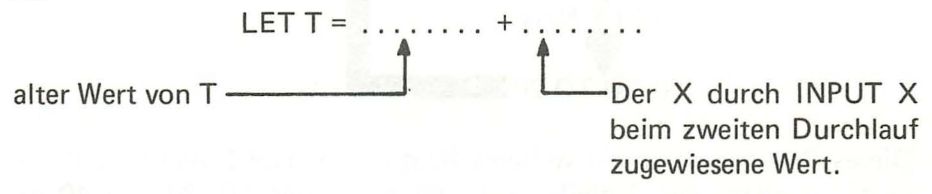
Wir wollen uns jetzt das Statement in Zeile 200 des Listings in Ab-

schnitt 45 etwas genauer ansehen. Beim ersten Lauf dieses Programms haben die Variablen, rechts vom Gleichheitszeichen in 200 LET T=T+X, den Wert:



Der neue Wert für T ist daher 12. Er wird vom Computer ausgedruckt. Beachten Sie, daß der Computer die gegenwärtigen Werte der Variablen, rechts vom Gleichheitszeichen, bei jeder erneuten Ausführung der Zeile 200 ersetzt.

Geben Sie die Werte für den zweiten Schleifendurchlauf des Programms an:



Der neue Wert von T ist ....

LET T = 12 + 43
55

48. Da Sie ja inzwischen wissen, wie man einen "durchgehenden" Computer anhalten kann, versuchen Sie es einmal mit dem folgenden Programm. Es veranlaßt den Computer, nacheinander alle ganzen Zahlen auszudrucken, also 1, 2, 3, 4, 5, 6 usw., bis Sie auf die Taste BREAK drücken.

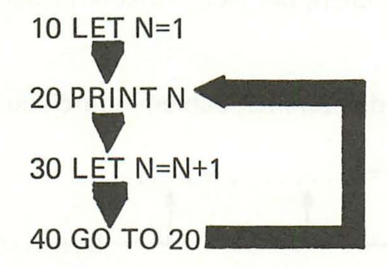
```
10 LET N=1
20 PRINT N
30 LET N=N+1
40 GO TO 20

RUN
```

1
2
3
4
5
6
7

STOPPED AT LINE 20

Wie arbeitet dieses Programm? Ganz einfach, folgen Sie den Pfeilen:



Dieses Programm ist ein weiteres Beispiel für eine Schleife. Zeile 10 liegt außerhalb der Schleife, während die Zeilen 20, 30 und 40 die Schleife bilden. Sie werden immer wieder durchlaufen. Zeile 10 wird nur einmal ausgeführt, wobei N mit dem Wert 1 initialisiert wird.

Zeile 30 liegt innerhalb der Schleife. Sie wird bei jedem Durchlauf ausgeführt. In Zeile 30 wird ein neuer Wert für N errechnet, indem 1 zum alten Wert für N addiert wird.

30 LET N = N + 1

neuer Wert (erstes N)
alter Wert (zweites N)

a) Nehmen Sie an, der alte Wert für N ist 1. Wie lautet der neue Wert für N? ....

b) Angenommen der alte Wert für N ist 2. Wie lautet dann der neue Wert von N? ....

\- -

a) 2 b) 3

49. Mit einer Zählschleife und einigen wenigen zusätzlichen Statements können wir ein Programm schreiben, das zeigt, wie unser Sparkapitel Jahr für Jahr anwächst. Hier ist es:

```
100 REM SEHEN SIE IHR GELD WACHSEN
110 PRINT "WENN SIE DIE HOEHE DES SPARKAPITALS"
120 PRINT "UND DEN ZINSFUSS PRO JAHR EINGEBEN,"
130 PRINT "WERDE ICH IHNEN ZEIGEN, WIE IHR GELD"
140 PRINT "JAHR UM JAHR ANWAECHST. UM MICH"
150 PRINT "ANZUHALTEN, DRUECKEN SIE DIE BREAK-TASTE."
155 PRINT
160 PRINT "SPARKAPITAL"
165 INPUT D
170 PRINT "ZINSFUSS";
175 INPUT R
180 LET N=1 <---------------- Anfangsjahr (N) = 1
190 PRINT
200 LET A=P*(1+R/100) ∧ N \
210 PRINT "JAHR=";N        }-> Berechne und drücke das Ergebnis für das Jahr N.
220 PRINT "BETRAG =";A    /
230 LET N=N+1 <-------------- Berechnung des nächsten Jahres
240 GOTO 190
```

```
RUN
WENN SIE DIE HOEHE DES SPARKAPITALS
UND DEN ZINSFUSS PRO JAHR EINGEBEN,
WERDE ICH IHNEN EZIGEN, WIE IHR GELD
JAHR UM JAHR ANWAECHST. UM MICH
ANZUHALTEN, DRUECKEN SIE DIE BREAK-TASTE

SPARKAPITAL? 1000
ZINSFUSS? 6

JAHR = 1
BETRAG = 1059.99999

JAHR = 2
BETRAG = 1123.59997
```

(In Kapitel 4 werden wir Ihnen zeigen, wie man das Ergebnis auf den nächsten Pfennig aufrundet.)

So läuft das Programm immer weiter, bis jemand auf die Taste BREAK drückt. Welche Statements sind Bestandteil der Schleife?
Geben Sie die Zeilennummern an. ....

– –

190, 200, 210, 220, 230, 240

Wir schlagen Ihnen auch vor, mit einem Filzschreiber ein Kästchen um die Schleife zu malen. Beachten Sie, daß die Zeilen 100 bis 180 außerhalb der Schleife liegen. Sie werden während eines Laufs nur einmal ausgeführt, während die Zeilen 190 bis 240 solange wiederholt werden, bis jemand auf BREAK drückt.

EIGENTEST

Arbeiten Sie jetzt bitte den nachfolgenden Eigentest durch, damit Sie feststellen können, wieviel Sie in diesem Kapitel gelernt haben.

1. Bevor wir ein Programm eingeben, schreiben wir zuerst NEW und betätigen dann die Taste RETURN. Warum? ....
   ....

2. Wie können wir den Computer dazu auffordern, ein vollständiges Listing eines in seinem Speicher befindlichen Programms auszugeben? ....

3. Nehmen wir einmal an, wir haben gerade ein Programm in den Computer eingegeben. Wie teilen wir ihm mit, daß er das Programm ausführen soll? ....

4. Nehme Sie an, das folgende Programm ist im Computer gespeichert.
   10 PRINT "MEIN COMPUTER VERSTEHT MICH"
   20 PRINT "MEIN COMPUTER VERWIRRT MICH"
   Beschreiben Sie, wie man das zweite Statement (Zeile 20) ersetzen

kann, ohne das gesamte Programm zu löschen. . . . .

. . . .

. . . .

5. Worin besteht der Unterschied zwischen direkten Statement und Statements, die im Speicher des Computers für spätere Ausführung abgelegt werden? . . . .

. . . .

6. Wir speichern das folgende Programm und lassen es ausführen.

```
100 PRINT "ICH BEIN EIN FLEISSIGER COMPUTER"
200 GO TO 100
```

Wie können wir den Computer anhalten? . . . .

7. Jedes der folgenden Statements enthält einen Fehler. Streichen Sie den Fehler an und schreiben Sie das Statement in korrekter BASIC-Syntax.

a) 20 PRINT DER RADUMFANG BETRAEGT;3.14*D . . . .

. . . .

b) 20 "DER RADUMFANG BETRAEGT";3.14*D . . . .

. . . .

c) 20 LET P*(1 + R/100)^N = A . . . .

. . . .

d) 20 TYPE "X=";X . . . .

. . . .

8. Beschreiben Sie, was bei jedem Statement falsch ist.

a) 99999 LET A = 7 . . . .

. . . .

b) 30 GO TO 3.14 . . . .

. . . .

c) 30 GO TO −100 . . . .

. . . .

d) 10 INPUT, Z . . . .

. . . .

e) 10 INPUT Z, ...

. . . .

9. Geben Sie an, was bei jedem Programm falsch ist.

a)
```
10 INPUT A          ..............................
20 INPUT B          ..............................
30 PRINT A+B        ..............................
40 GO TO 12         ..............................
```
b)
```
10 LET A = 7        ..............................
20 LET B = 5        ..............................
30 PRINT X + Y      ..............................
```

10. Vervollständigen Sie den Programmlauf auf der dafür vorgesehenen Zeile:

```
10 PRINT "A=";
15 INPUT A
20 PRINT "B=";
25 INPUT B
30 PRINT "A + B = ";A+B
RUN
A=? 7
B=? 5
```

....

11. Schreiben Sie ein Programm, mit dem sich Temperaturen, die in Grad Celsius angegeben sind, in Grad Fahrenheit umwandeln lassen. Benutzen Sie dazu die folgende Formel.

F = (9/5)C +32

Ein Lauf Ihres Programms sollte wie folgt aussehen:

```
RUN
SIE GEBEN DIE TEMPERATUR IN CELSIUSGRADEN EIN.
ICH GEBE IHNEN DIE TEMPERATUR IN FAHRENHEIT AN.

GRAD CELSIUS? 0
GRAD FAHRENHEIT = 32

GRAD CELSIUS? 100
GRAD FAHRENHEIT = 212

GRAD CELSIUS?
```

und so weiter ....

....

....
....
....
....
....
....
....

12. Glückwunsch! Sie sind der große Gewinner in einer Fernsehshow. Ihr Preis wird folgendermaßen ermittelt:
Zunächst wird eine Zufallszahl zwischen 1 und 1000 gezogen. Nennen wir sie N. Anschießend können Sie einen, aber nur einen der beiden folgenden Preise wählen, wozu Sie 60 Sekunden Zeit bekommen:
PREIS NR. 1: Sie erhalten N DM
PREIS NR. 2: Sie erhalten D DM, wobei $D = 1.01^N$ ist.

Vielleicht erkennen Sie die Formel für D wieder. Sie gibt den Betrag an, den man erhält, wenn man 1 DM bei 1%Zeinsen pro Tag anlegt, bei täglicher Berechnung in einem Zeitraum von N Tagen.
Es ergibt sich dabei natürlich folgende Frage: Welchen Preis soll man, bei gegebenem N wählen, Nr. 1 oder Nr. 2?
Schreiben Sie ein Programm, das eine sichere Entscheidung ermöglicht. Ein RUN des Programms sollte etwa so aussehen:

```
RUN

N=? 100
PREIS NR. 1 = 100
PREIS NR: 2 = 2.704781277

N=? 500
PREIS NR. 1 = 500
PREIS NR. 2 = 144.772491

N=? 1000
PREIS NR. 1 = 1000
PREIS NR. 2 = 20959.0741

N=?
usw.
```

Während Sie sich in den beiden ersten Fällen natürlich für Preis Nr. 1 entscheiden würden, wäre im Fall drei die Wahl von Preis 2 natürlich wesentlich besser.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

13. Schreiben Sie ein Programm, das Ihnen helfen soll, Ihren Kontostand zu überwachen. Hier ist ein RUN des gewünschten Programms:

```
RUN
ICH WERDE IHNEN HELFEN, IHREN KONTOSTAND ZU
UEBERWACHEN. GEBEN SIE AUSZAHLUNGEN UND SCHECKS
ALS NEGATIVE, EINZAHLUNGEN ALS POSITIVE
BETRAEGE AN.
ALTER KONTOSTAND? 123.45
EIN- ODER AUSZAHLUNG? –3.95
NEUER KONTOSTAND: 119.5
EIN- ODER AUSZAHLUNG? –25
NEUER KONTOSTAND: 94.5
EIN- ODER AUSZAHLUNG?
usw.
```

Erinnern Sie sich noch, wie man das Programm abrechen kann? Wenn nicht, sehen Sie noch einmal im Abschnitt 36 nach.

Hier ist Platz für Ihr Programm:

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

14. Nehmen Sie an, wir geben das nachfolgende Programm ein und starten es mit RUN. Geben Sie die ersten sieben Zahlen an, die vom Computer gedruckt werden.

```
10 LET N = 1
20 PRINT N
30 LET N = N + 1
50 GO TO 20
RUN
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Wie können wir den Computer daran hindern, noch weitere Zahlen zu drucken? . . . .

15. In den USA werden bekanntlich alle Maße auf das metrische System umgestellt. Schreiben Sie daher ein Programm, mit dem sich die Maße "Fuß" und "Zoll" in Zentimenter umwandeln lassen, wie es der folgende Programmlauf zeigt:

```
RUN

FUSS = ? 5
ZOLL = ? 11
ZENTIMETER = 180.34
```

Wir geben, wie gefordert, die Länge in Fuß und Zoll an. Der Computer berechnet dann die entsprechende Länge in Zentimetern und gibt sie aus.

Noch ein Beispiel:

```
FUSS = ? 3
ZOLL = ? 0
ZENTIMETER = 91.44

FUSS = ?
```

usw.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

ANTWORTEN ZUM EIGENTEST

Die Zahlen in den Klammern beziehen sich jeweils auf die Abschnitte, in denen der Stoff besprochen wurde.

1. Dadurch werden sämtliche alten Programme, die sich noch im Speicher des Computers befinden gelöscht. Wenn wir sie nicht löschen, kann es zu einem Durcheinander mit den Statements des neuen Programms kommen, was mit Sicherheit zu zahlreichen Fehlern führen würde. (Abschnitt 11, 12)

2. Schreiben Sie LIST und betätigen Sie die Taste RETURN (13,14)

3. Schreiben Sie RUN und betätigen Sie die Taste RETURN (14, 16)

4. Schreiben Sie eine neue Zeile mit der Nummer 20 und drücken Sie RETURN. Das alte Statement in Zeile 20 wird dadurch automatisch gelöscht und durch das neue ersetzt. Wenn wir nur die Zeilennummer einer Statements schreiben und RETURN betätigen, löscht der Computer das Statement mit dieser Zeilennummer, sofern in seinem Speicher überhaupt eins vorhanden war. (21)

5. Direkte Statements haben keine Zeilennummern und werden sofort nach dem betätigen von RETURN ausgeführt. Ein Statement, das gespeichert werden soll, muß eine Zeilennummer haben; der Computer erinnert sich dann bei der späteren Ausführung daran. (9)

6. Drücken Sie die Taste BREAK (34 – 36)

7\. a) 20 PRINT DER RADUMFANG BETRAEGT ;3.14*D
Die Anführungsstriche fehlen. Richtig:
20 PRINT "DER RADUMFANG BETRAEGT"; 3.14*D (42, 43)

b) 20 "DER RADUMFANG BETRAEGT" ;3.14*D
Die PRINT-Anweisung fehlt! Richtig:
20 PRINT "DER RADUMFANG BETRAEGT" ; 3.14*D (42)

c) 20 LET P*(1 + R/100) N = A
Auf der linken Seite von = darf nur eine Variable stehen! Richtig:
20 LET A = P*(1 + R/100)^N (Abschnitte 1 – 4)

d) 20 TYPE "X=";X
Der Computer weiß nicht, was TYPE bedeuten soll! Richtig:
20 PRINT "X=";X

8\. a) Zeilennumer zu groß (Abschnitt 9)

b) 3.14 ist keine gültige Zeilennummer, da sie nicht ganzzahlig ist und nicht im Bereich von 1 bis 65335 liegt. (Abschnitt 9)

c) –100 ist keine gültige Zeilennummer (Abschnitt 9)

d) dem Wort INPUT darf kein Komma folgen (Abschnitt 31)

e) Am Ende des INPUT-Statements darf kein Komma stehen (nur INPUT-Statements mit zwei oder mehr Variablen verwenden Kommas zwischen den Variablen) (Abschnitt 31)

9\. a) Im Programm ist die Zeilennummer 12 nicht vorhanden. Daher kann der Computer das Statement 40 GO TO 12 nicht ausführen. Stattdessen wird er ausgeben: ERROR– 12 IN ZEILE 40 (Abschnitt 33, 35, 36)

b) Das PRINT-Statement verwendet nicht die gleichen Variablen, denen in den ersten beiden Statements Werte zugeordnet wurden. (In ATARI BASIC würde der Computer in diesem Fall eine 0 für das PRINT-Statement ausgeben). Wir hätten das Programm auf eine der folgenden beiden Weisen schreiben müssen:

erste Möglichkeit	zweite Möglichkeit	
10 LET A = 7	10 LET X = 7	
20 LET B = 5	20 LET Y = 5	
30 PRINT A + B	30 PRINT X + Y	(1 – 6)

10. Hier ist der komplette Programmablauf:

```
RUN
A=? 7
B=? 5
A+B = 12
```

Das Statement: 30 PRINT "A+B=" ; A+B

veranlaßte: A+B = 12 (Abschnitt 42)

11.

```
100 REM UMWANDLUNG VON CELSIUS IN FAHRENHEIT
110 PRINT "SIE GEBEN DIE TEMPERATUR IN CELSIUSGRADEN
    EIN."
120 PRINT "ICH GEBE IHNEN DIE TEMPERATUR IN FAHRENHEIT
    AN."
130 PRINT
140 PRINT "GRAD CELSIUS";
150 INPUT C
160 PRINT "GRAD FAHRENHEIT = " ; (9/5)*C + 32
170 GOTO 130
```

(Abschnitte 42, 43)

12.

```
100 REM FERNSEHPREIS-PROBLEM
110 PRINT
120 PRINT "N=";
125 INPUT N
130 PRINT "PREIS NR. 1 =";N
140 PRINT "PREIS NR. 2 =";1.01   N
150 GOTO 110
```

(Abschnitte 42, 43)

13.

```
100 REM PROGRAMM ZUR KONTOFUEHRUNG
110 PRINT "ICH WERDE IHNEN HELFEN, IHREN KONTOSTAND
    ZU"
120 PRINT "UEBERWACHEN. GEBEN SIE AUSZAHLUNGEN UND
    SCHECKS"
130 PRINT "ALS NEGATIVE, EINZAHLUNGEN ALS POSITIVE"
140 PRINT "BETRAEGE AN.
150 PRINT
160 PRINT "ALTER KONTOSTAND";
165 INPUT B
```

```
170 PRINT
180 PRINT "EIN- ODER AUSZAHLUNG"
185 INPUT X
190 LET B=B+X
200 PRINT "NEUER KONTOSTAND:";B
210 GOTO 170
```

```
190 LET B = B + X
        ↑   ↑   ↑
        │   │   └──Ein- oder Auszahlung
        │   └──alter Kontostand
        └──neuer Kontostand
```

(Abschnitte 47 – 49)

14. RUN

```
1
2
3
4
5
6
7
usw.
```

Dieses Programm bildet eine Zählschleife. Sobald Sie RUN eingeben, beginnt der Computer zu zählen. Er wird solange weiterzählen, bis entweder einmal die Netzspannung ausfällt, ein Defekt auftritt oder jemand die Taste BREAK betätigt.(Abschnitt 47 – 49)

15.

```
100 REM UMWANDLUNG VON FUSS UND ZOLL IN ZENTIMETER
110 PRINT
120 PRINT "FUSS = ";
125 INPUT F
130 PRINT "ZOLL = ";
135 INPUT I
140 LET T = 12*F+1
150 LET C = 2.54*T
160 PRINT "ZENTIMETER =";C
170 GOTO 110
```

(Abschnitt 42)

NOTIZEN

Kapitel 4

Entscheidungen mit IF-THEN-Statements

Wahrscheinlich sind Sie inzwischen zu dem Schluß gekommen, daß ein Computer nur das ausführt, was Sie ihm ganz genau aufgetragen haben. Wie kann ein Computer daher jemals etwas "selbst" entscheiden? Nun, eigene Entscheidungen kann ein Computer sicherlich nicht treffen. Er ist aber in der Lage, gewisse Vergleiche vorzunehmen, deren Durchführung Sie ihm aufgetragen haben, und anschließend, je nachdem ob der von Ihnen spezifizierte Vergleich "wahr" oder "falsch" ausgefallen ist, im Programm entweder andere Instruktionen auszuführen oder Instruktionen zu überspringen. Das Statement, mit dem Sie derartige Vergleiche vorschreiben ist das IF-THEN-Statement. Wie Sie sehen werden, handelt es sich bei IF-THEN tatsächlich sogar um eine ganze "Familie" von Statements. Wenn Sie dieses Kapitel beendet haben, können Sie:

- das IF-THEN-Statement benutzen;
- die folgenden Vergleiche in einem IF-THEN-Statement benutzen:

Symbol	**Bedeutung**
⟨	kleiner als
⟩	größer als
=	gleich
⟨⟩	ungleich
⟩=	größer oder gleich
⟨=	kleiner oder gleich

- die RND-Funktion zur Erzeugung von Zufallszahlen benutzen;
- die INT-Funktion verwenden, um den Dezimalbruch-Anteil einer Zahl zu unterdrücken;
- Die INT- und RND-Funktionen zusammen benutzen, um Zufallszahlen zu erzeugen;
- Das ON-GOTO-Statement verwenden, um selektiv zu verschiedenen Statements in einem Programm verzweigen zu können;
- einen Doppelpunkt (:) verwenden, um mehrere Statements in der gleichen Zeile zu trennen (als Hilfsmittel zur Organisation Ihrer

Programme und um Speicherplatz einzusparen);

- Mehrfach-Statements in einer Zeile schreiben, um die Wirkung von IF-THEN-Statements noch zu steigern.

1. In diesem Kapitel wollen wir eine wichtige Fähigkeit von BASIC vorstellen, nämlich Vergleiche ausführen und Entscheidungen fällen zu können. Wir werden mit dem IF-THEN-(Wenn-Dann) Statement beginnen, das dem Computer aufträgt, eine spezifizierte Operation auszuführen, wenn (IF) eine bestimmte Bedingung "wahr" ist. Wenn die Bedingung jedoch nicht erfüllt, also "falsch" ist, wird die Operation nicht ausgeführt. Ein Beispiel für ein IF-THEN-Statement ist nachfolgend zu sehen:

```
150 IF X > 0 THEN PRINT "IHRE ZAHL IST POSITIV"
```

Das IF-THEN-Statement sagt dem Computer: Wenn der Wert von X größer als 0 ist, drucke die Meldung IHRE ZAHL IST POSITIV aus und fahre mit dem nächsten Statement in der Reihenfolge der Zeilennummern fort. Das Symbol > bedeutet "größer als".

Wenn der Wert von X kleiner oder gleich 0 ist, druckt der Computer die Mitteilung nicht aus. Stattdessen fährt er einfach mit der Ausführung des Programms fort.

Hier sehen Sie noch einmal, wie der Programmablauf durch Pfeile verdeutlicht werden kann:

Die Bedingung.

150 IF X > 0 THEN → PRINT "IHRE ZAHL IST POSITIV"

Folge diesem Pfad, wenn die Bedingung "falsch" ist.

Folge diesem Pfad, wenn die Bedingung "wahr" ist.

Wie lautet die Bedingung im obigen IF-THEN-Statement? ....
....

X > oder X größer als 0

2. Unter Verwendung des als Beispiels verwendeten Statements in Abschnitt 1 beantworten Sie bitte folgende Fragen:

a) Nehmen Sie an X = 3. Ist die Bedingung wahr oder falsch? ....

b) Nehmen Sie an X = —7. Ist die Bedingung wahr oder falsch? ....

c) Nehmen Sie an X = 0. Ist die Bedingung wahr oder falsch? ....

— —

a) Wahr. Der Computer druckt die Mitteilung IHRE ZAHL IST POSITIV aus.

b) Falsch. Der Computer druckt die Mitteilung nicht aus.

c) Falsch. Der Computer druckt die Mitteilung nicht aus.

3. Das Symbol > in einem IF-THEN-Vergleich bedeutet: "Ist größer als". Das Symbol = bedeutet "ist gleich". Sie können sich daher sicher leicht denken, welche Bedeutung das Symbol < in einem IF-THEN-Vergleich hat. ....

— —

(kleiner als)

4. Hier finden Sie ein einfaches Programm, das die Verwendung des IF-THEN-Statements verdeutlicht. In diesem Programm sind drei IF-THEN-Statements vorhanden, die dem Computer den Auftrag geben, "zu vergleichen und zu entscheiden". Denken Sie dabei daran, daß die Mitteilung in einem IF-THEN- PRINT-Statement nur ausgedruckt wird, wenn der Vergleich wahr ist.

```
100 REM***DIESES PROGRAMM ERMITTELT, OB X POSITIV,
    NEGATIV ODER NULL IST.
110 PRINT "WENN ICH SIE DAZU AUFFORDERE, GENEN SIE"
120 PRINT "BITTE EINE ZAHL EIN, UND ICH WERDE IHNEN
    SAGEN,"
130 PRINT "OB IHRE ZAHL POSITIV, NEGATIV ODER NULL IST."
135 PRINT
140 PRINT "WIE LAUTET IHRE ZAHL?";
145 INPUT X
150 IF X > 0 THEN PRINT "IHRE ZAHL IST POSITIV"
160 IF X < 0 THEN PRINT "IHRE ZAHL IST NEGATIV"
170 IF X = 0 THEN PRINT "IHRE ZAHL IST NULL"
180 GOTO 135
```

```
RUN
WENN ICH SIE DAZU AUFFORDERE, GEBEN SIE
BITTE EINE ZAHL EIN, UND ICH WERDE IHNEN SAGEN,
OB IHRE ZAHL POSITIV, NEGATIV ODER NULL IST.

WIE LAUTET IHRE ZAHL? -7
IHRE ZAHL IST NEGATIV

WIE LAUTET IHRE ZAHL? 3
IHRE ZAHL IST POSITIV

WIE LAUTET IHRE ZAHL? 0
IHRE ZAHL IST NULL

WIE LAUTET IHRE ZAHL? ■
```

Und so weiter. Sie wissen ja sicherlich noch, wie man ein Programm abbricht, das auf eine Eingabe wartet?

a) Wie lautet die Bedingung in Zeile 150? ....
b) Wie lautet die Bedingung in Zeile 160? ....
c) Wie lautet die Bedingung in Zeile 170? ....

– –

a) X 〉 0 oder X ist größer als 0
b) X 〈 0 oder X ist kleiner als 0
c) X = 0 oder X ist gleich 0

5. Beim Abarbeiten des Programms in Abschnitt 4 führt der Computer die Zeilen 100 bis 130 nur einmal aus, da sie außerhalb der Schleife liegen. Die Zeilen 135 bis 180 dagegen liegen innerhalb der Schleife und werden für jeden, vom Benutzer auf ein INPUT-Fragezeichen eingegebenen Wert für X ausgeführt. Nehmen wir einmal an, das Programm ist gestartet worden, der Benutzer tippt nach einem Fragezeichen die Zahl 13 ein und drückt dann die Taste RETURN. Dadurch wird der Wert 13 der Variablen X zugewiesen. Sehen Sie sich jetzt noch einmal das Programm an. Da X = 13, ist die Bedingung in Zeile 150 wahr, die Bedingungen in den Zeilen 160 und 170 sind dagegen falsch. Daher wird der Computer die Mitteilung in Zeile 150 ausdrucken, nicht aber die Mitteilungen in den Zeilen 160 und 170.

a) Angenommen X ist –7. Die Bedingung in Zeile 160 ist (wahr oder falsch) .... , und die Bedingungen in Zeile 150 und 170 sind

(wahr oder falsch) . . . .

b) Angenommen X ist 0. Welche Bedingung ist wahr? Geben Sie die Zeilennummer an: . . . .
Welche Bedingungen sind falsch? Geben Sie die Zeilennumern an: . . . .

— —

a) Wahr; falsch. Der Computer wird die Mitteilung in Zeile 160 ausdrucken, nicht aber die Mitteilungen in den Zeilen 150 und 170.

b) Zeile 170; Zeilen 150 und 160. Der Computer wird die Mitteilung in Zeile 170 ausdrucken, nicht aber die Mitteilungen in den Zeilen 150 und 160.

6. Das folgende Programm vergleicht zwei Zahlen A und B miteinander und druckt eine geeignete Mitteilung aus. Vervollständigen Sie die Zeilen 160 und 170 so, daß das Programm in der angegebenen Weise abläuft.

```
100 REM***VERGLEICHE ZWEI ZAHLEN, A und B
110 PRINT "GEBEN SIE WERTE FUER A UND B EIN"
120 PRINT
130 PRINT "A = ";
135 INPUT A
140 PRINT "B = ";
145 INPUT B
150 IF A > B THEN PRINT "A IST GROESSER ALS B"
160 IF A < B . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
170 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
180 GOTO 120
```

```
RUN
GEBEN SIE WERTE FUER A UND B EIN.

A = ? 1
B = ? 2
A IST KLEINER ALS B

A = ? 7
B = ? 2
A IST GROESSER ALS B

A = ? 55
B = ? 55
A IST GLEICH B
```

```
A = ?
```

```
160 IF A < B THEN PRINT "A IST KLEINER ALS B"
170 IF A = B THEN PRINT "A IST GLEICH B"
```

7. Wir wollen uns jetzt ein anderes IF-THEN-Statement ansehen.

Das Statement	190 IF X=–1 THEN GO TO 230
sagt dem Computer	Wenn der Wert von X gleich –1 ist, springe in Zeile 230. Wenn der Wert von X ungleich –1 ist, fährt der Computer mit seiner üblichen Zeilennummern-Reihenfolge fort.

Im allgemeinen hat das IF-THEN-Statement die folgende Form:

IF Bedingung THEN Statement

Das Statement kann nahezu jedes beliebige BASIC-Statement sein. Die Bedingung ist üblicherweise ein Vergleich zwischen einer Variablen und einer Zahl, oder zwischen zwei BASIC-Ausdrücken. Anschließend finden Sie eine Zusammenstellung der gebräuchlichen Vergleichssymbole.

BASIC-Symbol	**Vergleich**	**Math. Symbol**
=	gleich	$=$
<	kleiner als	$<$
>	größer als	$>$
<=	kleiner gleich	$\leq$
>=	größer gleich	$\geq$
<>	ungleich	$\neq$

Schreiben Sie jede der folgenden Bedingungen in einwandreiem BASIC:

a) M ist größer als 10: ....

b) Z ist kleiner gleich A zur zweiten Potenz: ....

c) X ist ungleich Y: ....

d) 3 mal P ist gleich Z mal Q: ....

---

a) M ⟩ 10
b) Z ⟨ = A ^ 2 oder Z ⟨ = A*A
c) X ⟨⟩ Y
d) 3 * P = Z * Q

8. Bis jetzt haben Sie zwei Mitglieder der Familie der IF-THEN-Statements kennengelernt und zwar:

IF-THEN PRINT (Mitteilung in Anführungszeichen)
IF-THEN GOTO (Zeilennummer)

Das zweite IF-THEN-Statement fordert den Computer auf, zu der angegebenen Zeilennummer zu "gehen" bzw. zu "verzweigen", sofern der Vergleich wahr ist. Sie können die Instruktion GOTO nach THEN auch weglassen und einfach die Zeilennummer angeben. Zeigen Sie, wie sich das folgende Statement auch kürzer schreiben läßt:

190 IF X = –1 THEN GOTO 230

....

– –

190 IF X = –1 THEN 230

9. Die grundlegende Form des IF-THEN-Statements lautete bekanntlich:

IF (Bedingung) THEN (nahezu jedes BASIC-Statement)

Da praktisch jedes BASIC-Statement auf THEN folgen darf, könnten wir den Computer beispielsweise veranlassen, einer Variablen einen Wert zuzuordnen oder durch INPUT nach einer Eingabe in einem IF-THEN-Statement zu fragen, sofern die Bedingung oder der Vergleich wahr sind.

IF Y = 3 THEN LET X = 1
IF X*X + Y*Y 25 THEN PRINT "IHR VERSUCH LIEGT RICHTIG"

Jetzt sollen Sie ein wenig üben, wie man IF-THEN-Statements schreibt, in denen Wertzuweisungen erfolgen, sofern der Vergleich wahr ist. Schreiben Sie für die folgenden Beispiele jeweils ein IF-THEN-Statement.

a) Wenn A ungleich B ist, weise der Variablen B den neuen Wert 10 zu. . . . .

b) Wenn H größer als 100 ist, soll der Computer ausgeben– "Hoppla! das sind zuviele Stunden!" . . . .

– –

a) IF A <> B THEN B = 10   Denken Sie daran, daß LET in einem Zuweisungs-Statement weggelassen werden kann.

b)
IF H > 100 THEN PRINT "HPPLA! DAS SIND ZUVIELE STUNDEN"

10. Schreiben Sie für jedes Beispiel ein IF-THEN-Statement.

a) Wenn der Wert von A kleiner oder gleich 10 ist, gehe zur Zeilennummer 100. . . . .

b) Wenn der Wert von A kleiner als 2*B ist, erhöhe den alten Wert von T um 1. . . . .

c) Wenn X größer als 2 mal Y ist, dann drucke die Mitteilung aus: "X ist mehr als doppelt so groß wie Y." . . . .

d) Wenn Z nicht gleich –1 ist, dann gebe einen neuen Wert für X ein. . . . .

– –

a) IF A <= 10 THEN 100   (GOTO kann weggelassen werden)
b) IF A < 2*B THEN T = T+1 oder
IF A < 2*B THEN LET T = T+1   (LET kann weggelassen werden)
c) IF X > 2*Y THEN PRINT "X IST MEHR ALS DOPPELT SO GROSS WIE Y"
d) IF Z <> –1 THEN INPUT X

11. Eine gebräuchliche Anwendung des IF-THEN-Statements besteht darin, ein als "Flag" bezeichnetes Signal zu erkennen, das einen Vorgang beendet und einem anderen einleitet. Hier finden Sie ein weiteres Beispiel für die teuerste Addiermaschine der Welt", die uns zum erstenmal in Abschnitt 45 von Kapitel 3 begegnete. Vielleicht ist es besser, wenn Sie sich diesen Abschnitt noch einmal ansehen, bevor Sie sich mit dem neuen Programm beschäftigen.

```
100 REM***DIE TEUERSTE ADDIERMASCHINE DER WELT
110 PRINT "ICH BIN DIE TEUERSTE ADDIERMASCHINE DER
    WELT."
120 PRINT "JEDESMAL, WENN ICH 'X=?' DRUCKE, GEBEN SIE"
130 PRINT "EINE ZAHL EIN. WENN SIE FERTIG SIND, SCHREI-
    BEN"
140 PRINT "SIE -1, UND ICH WERDE DIE SUMME IHRER ZUVOR"
150 PRINT "EINGEGEBENEN ZAHLEN AUSDRUCKEN."
160 T = 0
170 PRINT
180 PRINT "X=";
190 INPUT X
200 IF X = –1 THEN 230
210 T = T + X
220 GOTO 170
```

Die Zeilen 170 – 220 bilden eine GOTO-Schleife. Sobald jedoch jemand –1 als Wert für X eintippt, veranlaßt Zeile 200 den Computer, aus der Schleife herauszuspringen und zu Zeile 230 zu gehen.

```
230 PRINT
240 PRINT "SUMME ="; T
250 PRINT
260 GOTO 110
```

Die Zeilen 230 – 260 fordern den Computer auf, eine Leerzeile zu drucken, die Summe der Zahlen auszugeben und dann zur Zeile 110 zurückzuspringen und mit einem neuen Schleifendurchlauf zu beginnen.

```
RUN
ICH BIN DIE TEUERSTE ADDIERMASCHINE DER WELT.
JEDESMAL, WENN ICH 'X=?' DRUCKE, GEBEN SIE
EINE ZAHL EIN. WENN SIE FERTIG SIND, SCHREIBEN
SIE –1, UND ICH WERDE DIE SUMME IHRER ZUVOR
EINGEGEBENEN ZAHLEN AUSDRUCKEN.
X =? 6.95
X =? .47
X =? 8.49
X =? 3.06
X =? –1
```

Hier ist das von uns vorgesehene Flag, das dem Computer signalisiert: Das waren alle Zahlen.

```
SUMME=18.97
ICH BIN DIE TEUERSTE ADDIERMASCHINE DER WELT.
```

usw.

In unserem Programm wird –1 als Flag verwendet; das Statement 200 prüft jeden für X eingegebenen Wert. Sofern es sich dabei um –1 handelt, springt der Computer aus der Schleife heraus und zwar zur Zeile 230. Die Zeilen 230 bis 260 drucken die Summe T und veranlassen dann den Computer zur Zeile 110 zurückzuspringen und neu zu beginnen.

Jede beliebige Zahl, die nicht als INPUT-Wert vorkommt, könnte als Flag verwendet werden.

Modifizieren Sie das Programm bitte so, daß statt –1 die Zahl 999999 als Flag benutzt wird. Sie müssen dazu die Zeilen 140 und 200 ändern.

140 ....
200 ....

– –

```
140 PRINT "SIE 999999, UND ICH WERDE DIE SUMME IHRER
    ZUVOR"
200 IF X=999999 THEN 230
```

12. In dem Programm in Abschnitt 11 haben wir zuerst –1 in Flag verwendet. Das ist nur möglich, wenn keine der vorkommenden Zahlen negativ ist. In dem Fall empfiehlt sich die Verwendung des Flags 999999.

Mit einigen wenigen Änderungen können wir das Programm in Abschnitt 11 modifizieren, um den Mittelwert einer Folge von Zahlen zu berechnen. Die dazu verwendete Formel lautet:

$$\text{Mittelwert} = \frac{\text{Summe aller Zahlen}}{\text{Anzahl der Zahlen}} = \frac{T}{N}$$

In dem nachfolgenden Mittelwertprogramm verwenden wir die Variable T für die Summe der Zahlen und N für die eingegebene Anzahl. Vervollständigen Sie jetzt bitte das Programm:

```
100 REM***PROGRAMM ZUR MITTELWERTBERECHNUNG
110 PRINT "ICH BERECHNE DEN MITTELWERT MEHRERER
    ZAHLEN."
```

```
120 PRINT "WENN ICH 'X=?' AUSGEBE, TIPPEN SIE EINE ZAHL
    EIN."
130 PRINT "WENN SIE MIT DER EINGABE FERTIG SIND,
    SCHREIBEN SIE 999999."
140 LET T=0
150 LET N=. . . . . . . ◄—— Wir verwenden N zum Zählen der einge-
160 PRINT                   gebenen Zahlen.
170 PRINT "X= ";
175 INPUT X
180 IF X=999999 THEN 220
190 LET T=T+X
200 LET N=. . . . . . . ◄—— Erhöhe N um 1.
210 GOTO 160
220 PRINT
230 PRINT "N="; . . . . ◄—— Drucke zuerst die Anzahl der einge-
240 PRINT "SUMME = ";T      gebenen Zahlen aus.
250 PRINT "MITTELWERT = "; . . . . . . . . ◄—— Berechne den
260 PRINT                                      Mittelwert und
270 GOTO 110                                   drucke ihn aus.
```

Den kompletten Lauf des Programms sehen Sie nachstehend.

```
RUN
ICH BERECHNE DEN MITTELWERT MEHRERER ZAHLEN.
WENN ICH 'X=?' AUSGEBE, TIPPEN SIE EINE ZAHL EIN.
WENN SIE MIT DER AUFGABE FERTIG SIND, SCHREIBEN SIE
999999.
X =? 10

X =? –9

X =? –15

X =? –23

X =? –25

X =? –30

X =? 999999 ◄—————— Hier ist das Flag!

N = 7 ◄—————— N ist die Anzahl der eingegebenen Zahlen.

SUMME = –89
MITTELWERT = –12.71428571
```

- -

```
150 LET N=0
200 LET N=N+1
230 PRINT "N = "; N
250 PRINT "MITTELWERT = "; T/N
```

13. Die Wahl des Flags hängt sehr von der jeweiligen Problemstellung und den vorkommenden Daten ab. So kann bei verschiedenen Anwendungen auch 999999 als Flag völlig ungeeignet sein. Eine Zahl, die mit Sicherheit praktisch nie vorkommen dürfte, ist die Gleitkommazahl 1E97. Für die meisten Daten wird sie sich gut als Flag eignen. Modifizieren Sie daher bitte die Zeilen 130 und 180 im Programm aus Abschnitt 12 so, daß 1E97 als Flag verwendet wird.

130 ....

180 ....

- -

```
130 PRINT "WENN SIE MIT DER EINGABE FERTIG SIND,
    SCHREIBEN SIE 1E97."
180 IF X=1E97 THEN 220
```

14. Sie haben uns bis jetzt geholfen, eine Reihe von Programmen unter Verwendung von IF-THEN-Vergleichen zu schreiben. Jetzt dürfte es jedoch an der Zeit sein, daß Sie Ihren ersten "Alleinflug" machen und selbst ein Programm schreiben. Denken Sie gründlich über die Verwendung von IF-THEN-Vergleichen nach und erinnern Sie sich daran, was der Computer tut, wenn die Bedingungen "wahr" oder "falsch" sind. Anschließend finden Sie den ausgedruckten Lauf des Programms, das Sie schreiben sollen.

```
RUN
GEBEN SIE EINE ZAHL EIN, UND ICH WERDE IHNEN
SAGEN, OB SIE KLEINER ODER GLEICH 100 ODER
GROESSER ALS 100 IST.

IHRE ZAHL? 99
IHRE ZAHL IST 100 ODER KLEINER

IHRE ZAHL? 101
IHRE ZAHL IST GROESSER ALS 100
```

```
IHRE ZAHL IST GROESSER ALS 100
IHRE ZAHL? 100
IHRE ZAHL IST 100 ODER KLEINER
IHRE ZAHL?
```

Schreiben Sie jetzt das Programm.

```
100 REM***ZAHLENGROESSEN–PROGRAMM
110 PRINT "GEBEN SIE EINE ZAHL EIN, UND ICH WERDE IHNEN"
115 PRINT "SAGEN, OB SIE KLEINER ODER GLEICH 100 ODER"
120 PRINT "GROESSER ALS 100 IST."
130 PRINT
140 PRINT "IHRE ZAHL";
150 INPUT N
160 IF N <= 100 THEN PRINT "IHRE ZAHL IST 100 ODER
    KLEINER"
170 IF N > 100 THEN PRINT "IHRE ZAHL IST GROESSER ALS 100"
180 GOTO 130
```

15. Bald werden wir uns auch mit Computer-Spielen befassen, die viel Freude machen. Zuvor müssen Sie jedoch noch etwas über Zufallszahlen erfahren und die mit RND bezeichnete unvorhersagbare BASIC-Funktion kennenlernen.

Zufallszahlen sind Zahlen, die wahllos aus einer gegebenen Menge von Zahlen herausgegriffen werden. Viele Spiele verwenden beispielsweise einen Würfel oder einen Zahlenkreisel, um Zufallszahlen zu erzeugen.

Würfeln Sie eine 6, dürfen Sie sechs Stellen vorrücken.

Funktionen von BASIC sind automatisch ablaufende Vorgänge, die Sie zur Durchführung spezieller Operationen benutzen können. Diese Funktionen sind wie "eingebaute" Programme; die meisten von Ihnen könnten durch ein Programm oder ein Programm-Segment ersetzt werden. Da Computer jedoch häufig genug aufgefordert werden, die von diesen Funktionen bewirkten Operationen auszuführen, lohnt es sich, sie bereits in die Computer-Sprache einzubauen.

BASIC enthält eine spezielle Funktion mit der Bezeichnung RND, die Zahlen erzeugt, und zwar in scheinbar völlig willkürlicher Reihenfolge, so als wenn man sie aus einem Hut ziehen würde. Das nachfolgende Programm verdeutlicht die Verwendung der RND-Funktion.

Wir zeigen Ihnen hier das Programm und zwei verschiedene Programmläufe. Den ersten Lauf unterbrachen wir durch Eingabe von BREAK und starteten dann einen neuen.

```
10 PRINT RND ( 1 )
20 GOTO 10
```

```
RUN                          RUN

Ø.2772Ø642Ø8                 Ø.4468536376
Ø.Ø455932617                 Ø.2126617431
Ø.6158294677                 Ø.75856Ø18Ø6
Ø.Ø782165524                 Ø.5531ØØ5859
Ø.2481689453                 Ø.18786621Ø9
Ø.1948242187                 Ø.Ø785217285
Ø.96861267Ø8                 6.834Ø1722E-Ø3
Ø.5149536132                 Ø.2Ø4711914
Ø.8614654541                 STOPPED AT LINE 1Ø
Ø.382492Ø654

STOPPED AT LINE 1Ø
```

Wir haben BREAK gedrückt   Wir haben BREAK gedrückt

Sind die Listen der Zufallszahlen in den beiden Programmläufen gleich?

....

— —

Nein! Auch wenn Sie unser Programm in Ihren Computer eingeben und mit RUN starten, können Sie keineswegs eine der beiden Listen erwarten. Es sind eben wirklich rein zufällige Zahlen.

16. Das Statement 10 PRINT RND (1) veranlaßt den Computer, eine unterschiedliche Folge von Zahlen zu erzeugen, und zwar jedesmal, wenn das Programm abläuft. Die RND-Funktion liefert Zahlen, die rein willkürlich ausgewählt erscheinen. Bei unserem ATARI-Computer wird die RND-Funktion RND (1) geschrieben.

Wir setzen hier die Zahl 1 in Klammern hinter RND, jedoch kann auch jede andere positive Zahl verwendet werden, sogar Dezimalbrüche. Bei unserem Computer veranlaßt eine positive Zahl in Klammern hinter RND den Computer, bei jedem Programmlauf eine unterschiedliche Liste von Zufallszahlen zu erzeugen.

Sehen Sie sich die Zufallszahlen in Abschnitt 15 einmal genauer an und beantworten Sie bitte folgende Fragen:

a) Ist irgend eine Zahl negativ? ....
b) Ist irgend eine Zahl gleich Null? ....
c) Ist irgend eine Zahl größer als 1? ....
d) Ist irgend eine Zahl gleich 1? ....
e) Wie der Augenschein zeigt, sind die Zufallszahlen .... als Null und .... als 1.

— —

a) nein b) nein
c) nein d) nein
e) größer; kleiner

17. Es ist wahr, daß die von der RND-Funktion erzeugten Zufallszahlen größer als 0 und kleiner als 1 sind. Um es anders zu sagen: Von der RND-Funktion erzugte Zufallszahlen liegen zwischen 0 und 1. Oder in mathematischer Schreibweise:

$$0 < RND (1) < 1$$

Allerdings sind Zufallszahlen zwischen 0 und 1 nicht immer sehr zweckmäßig. Manchmal möchten wir auch die Ziffern von 0 bis 9, oder

ganzzahlige Zufallszahlen zwischen 1 und 100 haben. Nachstehend finden Sie ein Programm, in dem der Computer als Lernmaschine arbeitet, um Schülern die Addition einstelliger Zahlen beizubringen:

1 + 4 =? 5 ◄——————— Der Computer druckte 1+4=?
WEITER SO! RICHTIG — Der Schüler tippte 5 als Antwort ein und drückte auf RETURN.

9 + 6 =? 14
HMMM . . . FALSCH ◄——— Hier machte der Schüler einen Fehler.

9 + 6 =? 15 ◄——————— Der Computer wiederholt die Aufgabe.
WEITER SO! RICHTIG — Diesmal beantwortet der Schüler richtig.

8 + 0 =? — Eine neue Aufgabe . . . . . und so weiter

Unzweifelhaft sind Sie jetzt begierig darauf, das Programm zu sehen. Doch Geduld. Wir wollen es Stück für Stück zusammenbauen. Zunächst einmal müssen wir wissen, wie wir Zufalls-Zahlen erzeugen können.

Die von der RND-Funktion erzeugten Zufallszahlen sind gleichmäßig zwischen 0 und 1 verteilt. Das heißt, die Wahrscheinlichkeit für das Auftreten einer bestimmten Zahl ist für alle Zahlen gleich groß.

RND (1) liegt zwischen 0 und 1, ist aber niemals 0 oder 1. Daher liegt 10*RND(1) zwischen 0 und . . . .

– –

10; aber 10*RND(1) ist niemals gleich 0 oder 10.

18. Nachstehend sehen Sie ein Programm zum Ausdrucken von Zufallszahlen zwischen 0 und 10. Wir zeigen Ihnen zwei Läufe, um Sie daran zu erinnern, daß Sie jedesmal eine andere Liste von Zahlen erhalten.

```
10 PRINT 10*RND(1)
20 GOTO 10
```

```
RUN                 RUN

7.ØØ8Ø5664          1.17Ø19653
8.1372Ø7Ø3          5.948Ø2856
2.74749755          1.4836120Ø6
6.Ø7574462          3.89282225
3.9ØØ299Ø7          9.85137939
1.66ØØØ366          4.83612Ø61
4.94537259          2.Ø3781128
Ø.382461547
6.671295166
4.277496337         STOPPED AT LINE 1Ø

STOPPED AT LINE 1Ø
```

Jede Zufallszahl in beiden Läufen ist größer als 0 und kleiner als 10. Man kann sie sich aus einem ganzzahligen Anteil links vom Dezimalpunkt und einem gebrochenen Anteil rechts vom Punkt zusammengesetzt vorstellen.

Der ganzzahlige Anteil von 7.00805664 ist 7, der gebrochene Anteil .008050664

Der ganzzahlige Anteil von 0.382461547 ist Null (0), der gebrochene Anteil .38246547

a) Wie groß ist der ganzzahlige Anteil von 1.66000366? . . . .
Der gebrochene Anteil? . . . .

b) Wie groß ist der ganzzahlige Anteil von 4.83612061? . . . .
Der gebrochene Anteil? . . . .

---

a) 1; .66000366
b) 4; .83612061

19. Schreiben Sie jetzt drei kurze Programme, um Zufallszahlen in den jeweiligen angegebenen Bereichen zu erzeugen.

a) Zwischen 0 und 100 . . . .
. . . .

b) Zwischen 0 und 50 . . . .
. . . .

c) Zwischen 0 und 6 . . . .
. . . .

---

a) 10 PRINT 100*RND(1)
20 GOTO 10

```
RUN
73.33Ø8125
12.7318Ø66
37.9114539
87.8885Ø27
2.1Ø936ØØ1
2Ø.1256433
69.579Ø7Ø2
4.26Ø22968
```

b) 10 PRINT 50*RND(1)
20 GOTO 10

```
RUN
28.9513492
35.5718117
44.6Ø59Ø96
11.4345Ø26
6.48894357
15.2396ØØ4
```

c) 10 PRINT 6*RND(1)
20 GOTO 10

```
RUN
2.6ØØ18182
1.31444039
3.24Ø57255
Ø.1153584267
Ø.Ø2287558154
2.9Ø792334
5.721ØØ528
2.Ø32169Ø4
```

20. Für jede Zufallszahl zwischen 0 und 10 ist der ganzzahlige Anteil eine einzelne Ziffer. Wäre es da nicht viel einfacher, wir könnten den Computer gleich dazu veranlassen, den gebrochenen Anteil abzutrennen und nur gen ganzzahligen Anteil auszugeben?

Nun, wie Sie vielleicht schon vermuten werden, ist das möglich. BASIC verfügt nämlich über eine weitere, sehr nützliche Funktion, die INT genannt wird. Hier finden Sie dafür einige Beispiele:

INT(3) = 3	INT(7) = 7
INT(3.14159) = 3	INT(7.99999) = 7
INT(1.23456) = 1	INT(.999999) = 0

(Die Wirkungsweise der INT-Funktion kann in Worten so ausgedrückt werden: "Der ganzzahlige Anteil von 3.14159 ist 3.")

Allgemein gesagt, ist INT(X) der ganzzahlige Anteil von X. Wenden Sie jetzt einmal die Funktion INT auf einige der Zufallszahlen aus Abschnitt 18 an.

a) INT(7.00805664) = ...
b) INT(0.38246157) = ...
c) INT(2.74749755) = ...
d) INT(9.85137939) = ...

- -

a) 7 b) 0 c) 2 d)9

21. Vorsicht! INT arbeitet in der in Abschnitt 20 gezeigten Weise nur bei positiven Zahlen oder Null. Allgemein gesagt berechnet INT(X) die größte ganze Zahl, die kleiner oder gleich X ist. Zum Beispiel:

INT(3.14) = 3	aber INT(–3.14) = 4
INT(7) = 7	und INT(–7) = –7
INT(.999) = 0	aber INT(–.999) = –1

Für positive Zahlen oder 0 berechnet INT(X) den ganzzahligen Anteil von X. Beim Programmablauf wird der X zugeordnete Wert für die in den auf INT folgenden Klammern befindliche Variable X eingesetzt. Der Computer führt die INT-Funktion dann mit diesem numerischen Wert aus. Natürlich kann auch jede andere Variable an Stelle des Buchstabens X verwendet werden, sofern ihr zuvor im Programm ein Wert zugewiesen wurde.

Stellen Sie sich jetzt einmal vor, Sie sind der Computer. Geben Sie an, was Sie ausdrucken würden, wenn Ihr menschlicher Benutzer Sie dazu auffordert, die folgenden Programme auszuführen.

a)
```
10 X=15.77
20 PRINT INT(X)
RUN
```
....

b)
```
10 A=99.999
20 PRINT INT(A)
RUN
```
....

c)
```
10 F=98.6
20 PRINT INT(F)
RUN
```
....

- -

a) 15; b) 99; c) 98

22. Statt einer Zahl können wir auch eine Variable, eine Funktion oder jeden beliebigen BASIC-Ausdruck in die Klammern einsetzen, die auf das Wort INT folgen.

INT ( )
↑ Jede beliebige BASIC-Zahl, Variable, Funktion oder ein gültiger BASIC-Ausdruck kann hier eingesetzt werden.

Sie sollten mittlerweile zwischen Werten, Variablen, Ausdrücken, Funktionen und Strings unterscheiden können. Um zu zeigen, daß Sie die Bedeutungen kennen, geben Sie bei den folgenden Beispielen jeweils an, worum es sich handelt.

a) X ....
b) X*2 ....
c) RND(1) ....
d) "A+B IST EIN AUSDRUCK" ....
e) 22 ....
f) 1 + 1 ....
g) INT(52.88) ....
h) "=" ....
i) 4*(3+X) ....
j) Welches der obigen Beispiele könnte in den Klammern einer Funktion wie INT oder RND eingesetzt werden? ....

- -

a) Variable	b) Ausdruck	c) Funktion	d) String
e) Wert	f) Ausdruck	f) Funktion	h) String
i) Ausdruck	j) a, b, c, e, f, g, i		

23. Erinnern Sie sich daran: Die allgemeine Form der INT-Funktion lautet folgendermaßen:

INT (↑)

Wobei in den Klammern jede BASIC-Zahl, Variable, Funktion oder ein BASIC-Ausdruck stehen kann.

Es ist daher zulässig, wenn wir INT(10*RND(1)) schreiben. Dies ist ein Ausdruck, der eine Funktion als Bestandteil des Ausdrucks enthält.

RND(1) ist eine Zufallszahl zwischen 0 und 1.
10*RND(1) ist eine Zufallszahl zwischen 0 und 10.
INT(10*RND(1)) ist eine Zufallsziffer.

Das nachfolgende Programm veranlaßt den Computer, Zufallsziffern zu erzeugen und auszudrucken. Dabei handelt es sich um die Zahlen 0, 1, 2, 3, 4, 5, 6, 7, 8 und 9, also 0 bis 9 einschließlich. Beachten Sie, daß inklusive bedeutet, daß sowohl die Ziffern 0 und 9, als auch die dazwischenliegenden Ziffern eingeschlossen sind.

```
10 PRINT INT(10*RND(1))
20 GOTO 10
```

```
RUN                          RUN
5                            4
3                            6
2                            3
0                            9
6                            4
3                            0
4                            1
1                            2
7
STOPPED AT LINE 10           STOPPED AT LINE 10
```

Dieses Programm druckt Zufallsziffern, bei denen es sich um ganze Zahlen zwischen .... und .... einschließlich handelt.

0; 9

24. Schreiben Sie jetzt ein kurzes Programm, um ganzzahlige Zufallszahlen zwischen 0 und 19 zu erzeugen. Unser Programmlauf sieht folgendermaßen aus:

```
RUN
3
```

```
10
17
5
9
19
7
2
```

Schreiben Sie Ihr Programm in die beiden folgenden Zeilen:

. . . .

. . . .

– –

```
10 PRINT INT(20*RND(1))
20 GOTO 10
```

Bitte achten Sie darauf, daß Sie keine Klammer vergessen. Für jede linke Klammer muß eine rechte vorhanden sein, die sie wieder schließt. Wenn Sie das nicht berücksichtigen, liefert Ihnen der Computer eine Fehlermitteilung beim Programmlauf.

25. Wie erzielen wir ganzzahlige Zufallszahlen von 1 bis 20 anstatt 0 bis 19? Nun, das ist sehr einfach. Wir brauchen nur eine 1 zur Zufallszahl zu addieren, was auf zwei Arten möglich ist:

```
10 PRINT INT(20*RND(1)+1)
20 GOTO 10
```

oder

```
10 PRINT INT(20*RND(1))+1
20 GOTO 10
```

```
RUN
14
20
20
16
19
7
20
```

2
15

Die "+1" kann hinzuaddiert werden, bevor oder nachdem der ganzzahlige Anteil von 20*RND(1) ermittelt wurde. Die resultierende Zufallszahl ist in beiden Fällen gleich.

Schreiben Sie jetzt drei einfache Programme, um eine Liste von ganzzahligen Zufallszahlen auszudrucken.

a) von 1 bis 10 einschließlich

....

....

b) von 1 bis 100 einschließlich

....

....

c) von 1 bis 12 einschließlich

....

....

---

a)
```
10 PRINT INT(10*RND(1))+1
20 GOTO 10
```
oder
```
10 PRINT INT(10*RND(1)+1)
20 GOTO 10
```

b)
```
10 PRINT INT(100*RND(1))+1
20 GOTO 10
```
oder
```
10 PRINT INT(100*RND(1)+1)
20 GOTO 10
```

c)
```
10 PRINT INT(12*RND(1))+1
20 GOTO 10
```
oder
```
10 PRINT INT(12*RND(1)+1)
20 GOTO 10
```

26. Wie soll man vorgehen, wenn man Zufallszahlen von beispielsweise 5 bis 10 einschließlich wünscht? Hier ein kleiner Hinweis: Um Zufallszahlen von 1 bis 10 statt 0 bis 9 zu erhalten, haben wir + 1 addiert. Das kann man auch folgendermaßen darstellen:

0 bis 9
+1 +1
1 bis 10

a) Welche Zufallszahlen erzeugt die Funktion INT(6*RND(1))? ......

....

b) Welche Zahl müßen wir zu dem Ausdruck INT(6*RND(1)) hinzuaddieren, um ganze Zahlen von 5 bis 10 einschließlich zu erzeugen?

....

c) Welche Zahl müßen wir addieren, um Zufallszahlen zwischen 11 und 16 einschließlich zu erzeugen? ....

- -

a) 0 bis 5 einschließlich bedeutet: 0, 1, 2, 3, 4 und 5.

b) +5;
```
 0 bis 5
 +5  +5
 -------
 5   10
```

c) +11;
```
  0 bis 5
 +11  +11
 --------
 11   16
```

27. Sehen Sie sich jetzt noch einmal den Programmlauf in Abschnitt 17 an, bevor wir weitergehen. Jetzt sind wir dazu in der Lage das Programm zum Üben der Addition zusammenzubauen. Hier ist das erste Teilstück des Programms, das den RUN erzeugte.

```
100 REM***UEBUNGSPROGRAMM ZUR ADDITION
200 REM***ERZEUGE ZUFALLSZAHLEN A UND B
210 LET A=INT(10*RND(1))
220 LET B=INT(10*RND(1))
```

Für A und B werden folgende Werte erzeugt: ....

- -

die Zufallsziffern 0, 1, 2, 3, 4, 5, 6, 7, 8 oder 9

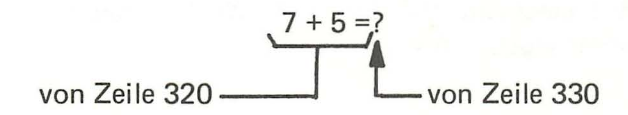
28. Sind Sie jetzt bereit für das nächste Teilstück? Also weiter!

```
300 REM***DRUECKE EINE AUFGABE UND FRAGE NACH
    ANTWORT
310 PRINT
320 PRINT A;" + ";B;" = ";
330 INPUT C
```

In Zeile 320 finden wir etwas Neues. Das Statement PRINT A;"+";B;" ="; fordert den Computer auf, den Wert von A, dann +, anschließend den Wert von B und schließlich = auszudrucken. Die einzelnen Bestandteile dieses Statements sind durch Semikolon voneinander getrennt. Das vierte Semikolon, ganz am Ende, sagt dem Computer, daß er nicht zur nächsten Zeile auf dem Bildschirm weitergehen soll. Mit anderen Worten es befiehlt dem Computer dort stehen zu bleiben, wo er mit dem Drucken aufhörte.

Erinnern Sie sich noch daran, daß das INPUT-Statement den Computer veranlaßt, ein Fragezeichen auszugeben? Dieses Fragezeichen wird in die gleiche Zeile wie die Information aus dem PRINT-Statement geschrieben, und zwar auf Grund des Semikolons am Ende von Zeile 320.

Ist beispielsweise A = 7 und B = 5, verursachen die Zeilen 320 und 330 folgende Ausgabe auf dem Bildschirm:



Wenn A = 3 ist und B = 4, was wird dann durch die Statements in den Zeilen 320 und 330 ausgedruckt? ....

---

3 + 4 = ?

29. Nachdem der Schüler eine Antowrt eingegeben und die Taste RETURN betätigt hat, weist der Computer diesen Wert der Variablen C zu und fährt fort.

Bitte beachten Sie auch, wie wir in diesem Programm REM-Statements verwenden, um etwas über den Inhalt jedes Programm-Teilstücks zu sagen. Das ist zwar für den Computer bedeutungslos, hilft aber dem Leser des Programms, seine Arbeitsweise besser zu verstehen.

```
400 REM***IST DIE ANTWORT RICHTIG?
410 IF C=A+B THEN 610
```

Wenn die Antwort (C) des Schülers richtig ist, geht der Computer

weiter zur Zeile ....

----

610 (Ist die Antwort falsch, fährt der Computer im normalen Programmablauf fort.)

30. Wenn die Antwort des Schülers nicht richtig ist, war die IF-THEN Bedingung falsch. Der Computer führt als nächstes folgende Statements aus, während er das Programm in der Reihenfolge der Zeilennummern abarbeitet:

```
500 REM***DIE ANTWORT IST FALSCH
510 PRINT "HMMM ... ICH HABE EIN ANDERES ERGEBNIS."
521 GOTO 310
```

Nehmen wir einmal an, die Antwort ist falsch. Der Computer druckt dann "HMMM . . . ICH HABE EIN ANDERES ERGEBNIS und geht weiter zur Zeile 310. Was geschieht als nächstes? (Wiederholen Sie evtl. noch einmal die vorangegangenen Abschnitte.) ....
....

Der Computer wiederholt die Aufgabe.

31. Sehen Sie sich noch einmal Abschnitt 29 an, der die Zeilen 400 und 410 enthält. Sofern die Antwort des Schülers richtig war, veranlaßt Zeile 410 den Computer zur Zeile 610 weiterzugehen.

```
600 REM***DIE ANTWORT IST RICHTIG
610 PRINT "WEITER SO! RICHTIG BEANTWORTET!"
620 GOTO 210
```

Nehmen wir an, die Antwort ist korrekt. Der Computer druckt WEITER SO! RICHTIG BEANTWORTET! und geht dann weiter zur Zeile 210. Was geschieht als nächstes? ....

Der Computer erzeugt eine neue Aufgabe (neue Werte für A und B) und druckt sie aus.

32. Nachstehend finden Sie das Listing des vollständigen Übungsprogramms.

```
100 REM***UEBUNGSPROGRAMM ZUR ADDITION
200 REM***ERZEUGE ZUFALLSZAHLEN A UND B
210 LET A=INT(10*RND(1))
220 LET B=INT(10*RND(1))
300 REM***DRUCKE EINE AUFGABE UND FRAGE NACH DER
    ANTWORT
310 PRINT
320 PRINT A; " + "; B; " = ";
330 INPUT C
400 REM***IST DIE ANTWORT RICHTIG?
410 IF C=A+B THEN 610
500 REM***DIE ANTWORT IST FALSCH
510 PRINT "HMMM . . . ICH HABE EIN ANDERES ERGEBNIS."
520 GOTO 310
600 REM***DIE ANTWORT IST RICHTIG
610 PRINT "WEITER SO! RICHTIG BEANTWORTET!
620 GOTO 210
```

Ändern Sie Zeile 210 so, daß der Wert von A eine ganzzahlige Zufallszahl zwischen 0 und 19 ist, statt 0 bis 9.

210 LET A= . . . .

— —

210 LET A=INT(20*REND(1))

33. Seien Sie bei der nächsten Aufgabe vorsichtig! Ändern Sie bitte Zeile 220 so, daß der Wert von B eine ganzzahlige Zufallszahl zwischen 10 und 20 einschließlich ist.

220 LET B= . . . .

— —

220 LET B=INT(11*RND(1))+10

Damit erhalten wir die Zahlen 10 bis 20 einschließlich, also insgesamt 11 Stück.

```
  0 bis 10
+10    +10
----------
 10 bis 20
```

Ein Lauf mit den beiden genänderten Programmzeilen 210 und 220 lieferte folgenden Ausdruck:

```
RUN
6 + 14 =? 20
WEITER SO! RICHTIG BEANTWORTET!

4 + 10 =? 15
HMMM . . . ICH HABE EIN ANDERES ERGEBNIS.

4 + 10 =? 14
WEITER SO! RICHTIG BEANTWORTET!

6 + 16 =?
```

Und so weiter!

34. Wenn die Antwort des Schülers richtig ist, druckt der Computer immer aus: WEITER SO! RICHTIG BEANTWORTET! Das ist ein bißchen eintönig und um diese Monotonie zu beseitigen, wollen wir das Programm so modifizieren, daß der Computer jeweils wahllos aus drei möglichen Antworten bei einem richtigen Ergebnis eine auswählt. Die Änderungen erfolgen in dem Programmbereich, der mit der Zeile 600 beginnt.

```
600 REM***DIE ANTWORT IST RICHTIG
610 LET R=INT(3*RND(1))+1  <———— Beachten Sie die +1
620 IF R=1 THEN 630
621 IF R=2 THEN 650
622 IF R=3 THEN 670
630 PRINT "WEITER SO! RICHTIG BEANTWORTET!"
640 GOTO 210
650 PRINT "SIE HABEN ES GEWUSST! MACHEN SIE NOCH
    EINEN VERSUCH!"
660 GOTO 210
670 PRINT "TOLL HABEN SIE DAS GEMACHT! MACHEN SIE SO
    WEITER!"
680 GOTO 210
```

Die möchlichen Werte von R sind . . . .

— —

1, 2 und 3 (Nicht 0, 1 und 2, da wir +1 am Ende der Zeile 610 addiert haben).

35. Wir sehen also, daß R nur 1, 2 oder 3 sein kann.

a) Was druckt der Computer bei R=1? ....
b) Was druckt der Computer bei R=3? ....
c) Was druckt der Computer für R=2? ....

— — — — — — — — — — — — — — — — — — — —

a) WEITER SO! RICHTIG BEANTWORTET!
b) TOLL HABEN SIE DAS GEMACHT! MACHEN SIE SO WEITER!
c) SIE HABEN ES GEWUSST! MACHEN SIE NOCH EINEN VERSUCH!

36. Die Änderungen aus Abschnitt 34 haben wir in das ursprüngliche Programm aus Abschnitt 32 eingebaut und dann einen Lauf des modifizierten Programms gestartet. Hier sehen Sie was geschah:

```
RUN

9 + 7 =? 16
SIE HABEN ES GEWUSST! MACHEN SIE NOCH EINEN VERSUCH!

6 + 3 =? 9
TOLL HABEN SIE DAS GEMACHT! MACHEN SIE SO WEITER!

5 + 9 =? 14
WEITER SO! RICHTIG BEANTWORTET!
```

Wenn die Antwort des Schülers falsch ist, druckt der Computer immer nur HMMM . . . ICH HABE EIN ANDERES ERGEBNIS. Modifizieren Sie daher bitte das Programm in Abschnitt 32 so, daß der Computer, bei einer falschen Antwort, zufällig einen der beiden folgenden Sätze auswählt:

HMMM . . . ICH HABE EIN ANDERES ERGEBNIS.
VERSUCHEN SIE ES MIT EINEM ANDEREN ERGEBNIS.

```
500 LET R= ..................................
520 IF ......................................
530 IF ......................................
540 PRINT ...................................
550 GOTO 310
560 .........................................
570 GOTO 310
```

— — — — — — — — — — — — — — — — —

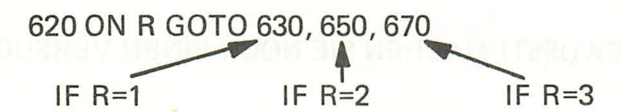
```
500 REM***DIE ANTWORT IST FALSCH
```

```
510 LET R=INT(2*RND(1))+1
520 IF R=1 THEN 540
530 IF R=2 THEN 560
540 PRINT "HMMM . . . ICH HABE EIN ANDERES ERGEBNIS."
550 GOTO 310
560 PRINT "VERSUCHEN SIE ES MIT EINEM ANDEREN
    ERGEBNIS."
570 GOTO 310
```

37. Die drei Statements:

```
620 IF R=1 THEN 630
621 IF R=2 THEN 650
622 IF R=3 THEN 670
```

können durch ein einzelnes Statement ersetzt werden.



Mit dieser Änderung kann das Programm-Segment in Abschnitt 34 wie folgt umgeschrieben werden.

```
600 REM***DIE ANTWORT IST RICHTIG
610 LET R=INT(3*RND(1))+1
620 ON R GOTO 630, 650, 670
630 PRINT "WEITER SO! RICHTIG BEANTWORTET!"
640 GOTO 210
650 PRINT "SIE HABEN ES GEWUSST! MACHEN SIE NOCH EINEN
    VERSUCH!"
660 GOTO 210
670 PRINT "TOLL HABEN SIE DAS GEMACHT! MACHEN SIE SO
    WEITER!"
680 GOTO 210
```

a) Sehen Sie sich Zeile 610 einmal an. Welche Werte kann R annehmen?

. . . .

b) Nehmen wir an, während eines Laufs wird von RND(1) in Zeile 610 die Zufallszahl .34319 erzeugt. Welchen Wert erhält R dann? . . . . . .

. . . .

c) In welche Zeile wird das "ON R GOTO"-Statement in diesem Fall den Computer schicken? . . . .

---

a) 1, 2, 3
b) 2, INT(3*.34319)+1 = INT(1.02957)+1 = 1 + 1 = 2
c) Zeile 650

38. Allgemein hat das ON . . . GOTO-Statement die folgende Form:

$$\text{ON e GOTO } \ell_1, \ell_2, \ell_3, \ldots, \ell n$$

Darin kann e jeder beliebige BASIC-Ausdruck sein, ℓ_1, ℓ_2 usw. bis ℓ sind Zeilennummern. Gültige Werte für e sind die ganzen Zahlen 1, 2, 3, . . . n. Für e = 1 geht der Computer zur Zeile ℓ_1. Für e = 2 geht er zur Zeile ℓ_2. Und so weiter.

Verwenden Sie ein ON . . . GOTO-Statement, um die beiden IF-Statements in den Zeilen 520 und 530 unseres Programms aus Abschnitt 36 zu ersetzen.

```
500 REM***DIE ANTWORT IST FALSCH
510 LET R=INT(2*RND(1))+1
520 ..............................................
540 PRINT "HMMM . . . ICH HABE EIN ANDERES ERGEBNIS."
550 GOTO 310
560 PRINT "VERSUCHEN SIE ES MIT EINEM ANDEREN
    ERGEBNIS!"
570 GOTO 310
```

```
520 ON R GOTO 540, 560
```

39. Jetzt werden wir eine wunderbare Methode kennenlernen, um Platz zu sparen. Hier finden Sie den ersten Teil unseres Programms zur Übung der Addition, und zwar in einer Version mit mehreren Statements je Zeile.

```
100 REM***UEBUNGSPROGRAMM ZUR ADDITION
200 REM***ERZEUGE ZUFALLSZAHLEN A UND B
210 LET A=INT(20*RND(1))
220 LET B=INT(11*RND(1))+10
300 REM***DRUCKE EINE AUFGABE UND FRAGE NACH DEM
    ERGEBNIS
```

```
310 PRINT:PRINT A;"+";B;"="; : INPUT C
400 REM***IST DIE ANTWORT RICHTIG?
410 IF C=A+B THEN 600
500 REM***DIE ANTWORT IST FALSCH
510 LET R=INT(2*RND(1))+1
520 ON R GOTO 540, 560
540 PRINT "HMMM . . . ICH HABE EIN ANDERES ERGEBNIS."
    : GOTO 310
560 PRINT "VERSUCHEN SIE ES MIT EINEM ANDEREN
    ERGEBNIS!" : GOTO 310
```

In diesem Programm enthält die Zeile 310 drei Statements, während die Zeilen 540 und 560 jeweils zwei Statements enthalten. Welches Symbol wird in einer Zeile, die mehr als ein Statement enthält, zwischen den einzelnen Statements verwendet? ....

– – – – – – – – – – – – – – – – – –

ein Doppelpunkt (:)

```
310 PRINT : PRINT A; " + " ;B; " = "; : INPUT C
```

40. Zur besseren Lesbarkeit sehen wir gewöhnlich zu beiden Seiten des Doppelpunktes einen Zwischenraum vor. Das ist jedoch nicht notwendig.

Ob wir Zwischenräume haben oder nicht, der Computer versteht die Statements so oder so.

Im letzten Abschnitt haben wir Ihnen nicht das ganze Programm gezeigt, da wir gern möchten, daß Sie den mit Zeile 600 im Programmbeispiel in Abschnitt 37 beginnenden Programmteil umschreiben, und zwar unter Verwendung von Mehrfachstatements in den Zeilen, in denen es sinnvoll ist, mehrere Statements zusammenzufassen.

```
600 REM***DIE ANTWORT IST RICHTIG
610 LET R=INT(3*RND(1))+1
620 ON R GOTO 630, 640, 650
630 ..........................................
640 ..........................................
650 ..........................................
```

```
– – – – – – – – – – – – – – – – – – – – – –
630 PRINT "WEITER SO! RICHTIG BEANTWORTET!" : GOTO 210
640 PRINT "SIE HABEN ES GEWUSST! MACHEN SIE NOCH EINEN
    VERSUCH!" : GOTO 210
650 PRINT "TOLL HABEN SIE DAS GEMACHT! MACHEN SIE SO
    WEITER!" : GOTO 210
```

41. Hier ist jetzt endlich das Computerspiel, das wir Ihnen bereits früher in diesem Kapitel versprochen hatten. Beachten Sie die Verwendung von Mehrfach-Statements in den Zeilen 150, 160, 170 und 180.

```
100 REM***ERRATEN SIE MEINE ZAHL
110 LET X=INT(100*RND(1))+1
120 PRINT
130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100."
140 PRINT "ERRATEN SIE MEINE ZAHL!"
150 PRINT : PRINT "IHR VERSUCH" ; : INPUT G
160 IF G < X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL." : GOTO 150
170 IF G > X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    KLEINEREN ZAHL." : GOTO 150
180 IF G = X THEN PRINT "BRAVO! SIE HABEN MEINE ZAHL
    ERRATEN!" : GOTO 110

RUN

ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100.
ERRATEN SIE MEINE ZAHL!

IHR VERSUCH? 50
VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL.

IHR VERSUCH? 75
VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL.

IHR VERSUCH? 73
BRAVO! SIE HABEN MEINE ZAHL ERRATEN!

ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 100.
ERRATEN SIE MEINE ZAHL!

IHR VERSUCH?
```

Und so weiter.

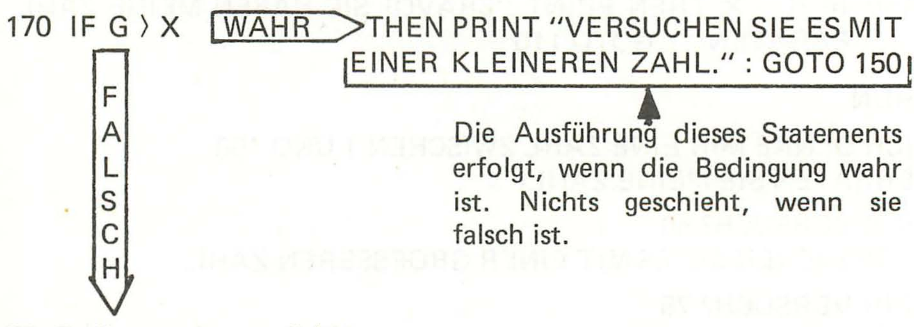
Zeile 110 fordert den Computer auf, eine Zufallszahl zu erzeugen und sie in der Variablen X zu speichern. Diese Zufallszahl ist eine .... zwischen .... und ....

---

ganze Zahl; 1; 100

42. Mehrfach-Statements in einer Zeile ermöglichen uns eine weitere angenehme Abkürzung. Sie erinnern sich sicher daran, daß der Computer, sofern der Vergleich in einem IF-THEN-Statement falsch ist, den Rest des Statements überspringt und zur nächsten Zeile des Programms in der Reihenfolge der Zeilennummern weitergeht. Das schöne daran ist, daß der Computer auch alle anderen Statements, die einem falschen IF-THEN-Vergleich folgen überspringt, sofern Sie sich in der gleichen Zeile befinden. Die restlichen Statements dieser Zeile werden nur ausgeführt, wenn der Vergleich im IF-THEN-Statement wahr ist. Sie können den Computer mit einer Zeile also dazu veranlassen, mehr als nur eine Operation auszuführen, sofern der Vergleich war ist.

Sehen Sie sich dazu noch einmal die Zeile 170 an, so wie sie nachfolgend aufgeführt ist. Der Computer wird sowohl PRINT als auch GOTO ausführen, sofern die Bedingung wahr ist, aber er tut keins von beiden, wenn die Bedingung falsch ist.



Die Bedingung lautet: G ⟩ X

a) Nehmen Sie an, G = 90 und X = 73. Ist die Bedingung WAHR oder FALSCH? ....
b) Nehmen Sie an, G = 70 und X = 73. Ist die Bedingung WAHR oder FALSCH? ....
c) Nehmen Sie an, G = 73 und X = 73. Ist die Bedingung WAHR oder

FALSCH? ....

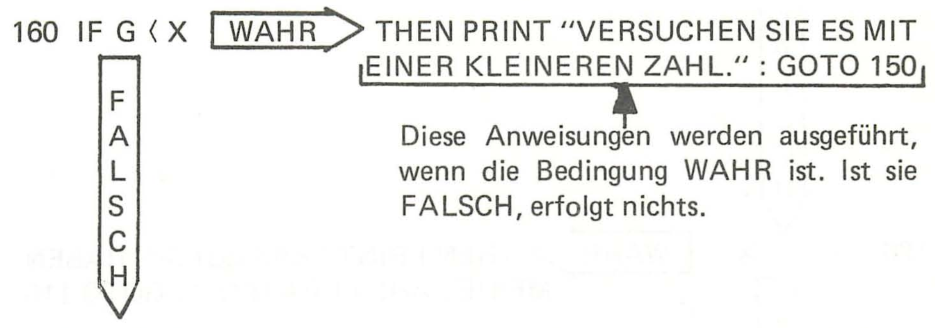
---

a) WAHR; b) FALSCH; c) FALSCH

43. Zeile 150 veranlaßt den Computer "IHR VERSUCH?" auszudrucken. Wenn der Spieler einen Versuch eintippt, speichert der Computer ihn in der Variablen G. Die Zeilen 160, 170 und 180 vergleichen den Versuch G mit der Zufallszahl X. Sehen wir uns Zeile 160 einmal an:

```
160 IF ‹ X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL." : GOTO 150
```

Wenn der Versuch G kleiner als die Zufallszahl X ist, druckt der Computer die Mitteilung VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL aus und geht dann (GOTO) zur Zeile 150, um nach einem neuen Versuch zu fragen. Wenn jedoch G größer oder gleich X ist, werden weder die PRINT- noch die GOTO-Anweisung ausgeführt.



Ist daher G ‹ X FALSCH, geht der Computer zur Zeile 170. Beschreiben Sie, was geschieht, wenn er Zeile 170 ausführt.

a) Wenn G größer als X ist (G › X ist WAHR), dann ....

....

b) Wenn jedoch G nicht größer als X ist (G › X ist FALSCH), dann ....

....

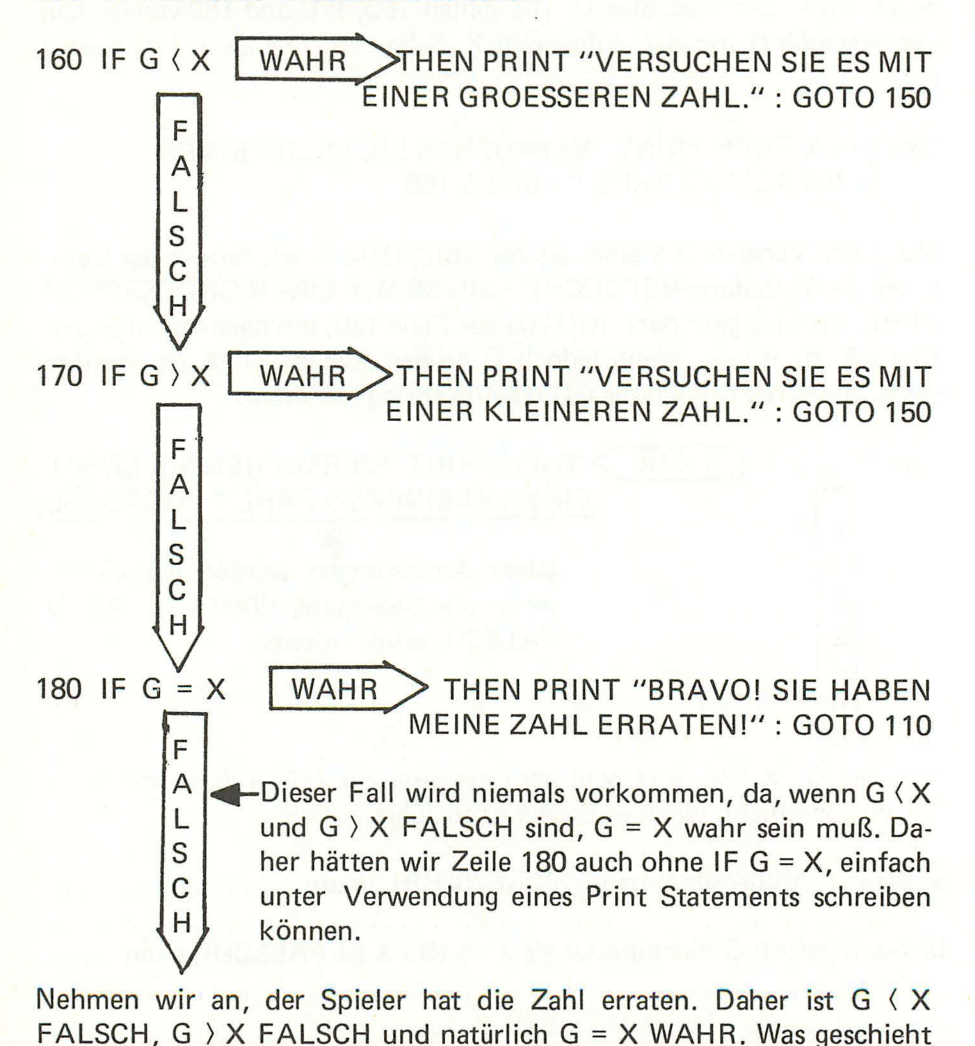
---

a) druckt der Computer die Mitteilung VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL und geht anschließend zur Zeile 150 für

einen weiteren Versuch.

b) werden weder die PRINT- noch die GOTO-Anweisung ausgeführt.

44. Wenn G ⟨ X FALSCH und G ⟩ X FALSCH sind, wird der Computer schließlich zur Zeile 180 gelangen. Dieser Vorgang ist anschließend noch einmal etwas verdeutlicht zu sehen:



Nehmen wir an, der Spieler hat die Zahl erraten. Daher ist G ⟨ X FALSCH, G ⟩ X FALSCH und natürlich G = X WAHR. Was geschieht jetzt? . . . .

. . . .

– –

Der Computer druckt den Text BRAVO! SIE HABEN MEINE ZAHL ERRATEN, geht dann zur Zeile 110, wo er angewiesen wird, sich eine neue Zahl auszudenken und beginnt mit einem neuen Spiel.

45. Für sehr junge Spieler kann es vielleicht wünschenswert sein, den Bereich der durch das Statement mit der RND-Funktion (Zeile 110) erzeugen ganzen Zahlen einzuschränken. Statt Zahlen zwischen 1 und 100 könnten wir beispielsweise nur Zahlen zwischen 1 und 25 vorsehen. Umgekehrt werden erfahrene Spieler vielleicht einen größeren Zahlenbereich haben wollen, beispielsweise 1 bis 1000.

a) Modifizieren Sie die Zeilen 110 und 130 so, daß der Zahlenbereich zwischen 1 und 25 liegt:
   110 ....
   130 ....
b) Modifizieren Sie die Zeilen 110 und 130 so, daß der Zahlenbereich jetzt zwischen 1 und 1000 liegt:
   110 ....
   130 ....

– –

a) 110 LET X=INT(25*RND(1))+1
   130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 25."
b) 110 LET X=INT(1000*RND(1))+1
   130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND 1000."

46. Hier finden Sie einen bequemen Weg zur Änderung des Zahlenbereichs:

```
100 REM***ERRATEN SIE MEINE ZAHL
105 LET R=100
110 LET X=INT(R*RND(1))+1
120 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND"; R
```

Der Zahlenbereich wird jetzt durch 1 und R festgelegt, wobei R in Zeile 105 ein Wert zugewiesen wird, der in den Zeilen 110 und 130 benutzt wird. Um den Zahlenbereich zu verändern, brauchen Sie jetzt nur noch die Zeile 105 ändern. Angenommen, Sie wollen Zahlen im Bereich von

1 bis 500 haben, was müssen Sie dann in Zeile 105 schreiben?

105 ....

— —

105 LET R=500 oder einfach 105 R=500

47. Modifizieren Sie, durch Verwendung von IF-THEN-Statements und Zeilen mit Mehrfach-Statements, das ZAHLENGROESSEN-Programm aus Abschnitt 14 dieses Kapitels derart, daß ein Lauf so aussieht, wie nachfolgend angegeben:

```
RUN
GEBEN SIE EINE ZAHL EIN, UND ICH WERDE IHNEN SAGEN, OB
SIE KLEINER ALS 100 IST, ZWISCHEN 100 UND 1000 LIEGT,
ODER GROESSER ALS 1000 ist.

IHRE ZAHL? 99
IHRE ZAHL IST KLEINER ALS 100.

IHRE ZAHL? 999
IHRE ZAHL LIEGT ZWISCHEN 100 UND 1000.

IHRE ZAHL? 1001
IHRE ZAHL IST GROESSER ALS 1000.

IHRE ZAHL?
```

....

....

....

....

....

....

....

....

....

....

— —

```
100 REM***NEUES ZAHLENGROESSEN-PROGRAMM
110 PRINT "GEBEN SIE EINE ZAHL EIN, UND ICH WERDE IHNEN
    SAGEN, OB
120 PRINT "SIE KLEINER ALS 100 IST, ZWISCHEN 100 UND
```

```
     1000 LIEGT,"
125 PRINT "ODER GROESSER ALS 1000 IST."
130 PRINT : PRINT "IHRE ZAHL"; : INPUT N
140 IF N 1000 THEN PRINT "IHRE ZAHL IST KLEINER ALS
     100." : GOTO 130
150 IF N = 1000 THEN PRINT "IHRE ZAHL LIEGT ZWISCHEN
     100 UND 1000." : GOTO 130
160 IF N 1000 THEN PRINT "IHRE ZAHL IST GROESSER ALS
     1000." : GOTO 130
```

48. Wir möchten gern, daß sie jetzt Ihre bisher gesammelten Kenntnisse in BASIC dazu verwenden, ein Programm auszuarbeiten und zu schreiben, das es ermöglicht, das Einmaleins von 0 x 0 bis 12 x 12 zu erlernen, bzw. zu wiederholen. Wir wollen dieses Programm MULTIPLIKATIONS-UEBUNGS-PROGRAMM nennen. Sehen Sie sich den Programmlauf an und bauen Sie dann Ihr Programm zusammen. Wenn Sie einen Computer zur Verfügung haben, überprüfen Sie bitte Ihr eigenes Programm, bevor Sie sich unsere Lösung ansehen.

Vergleichen Sie die jeweilige Lösung des Spielers mit der richtigen Lösung.

Wenn die Antwort des Spielers kleiner als das korrekte Ergebnis ist, drucken Sie VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL und wiederholen die Aufgabe.

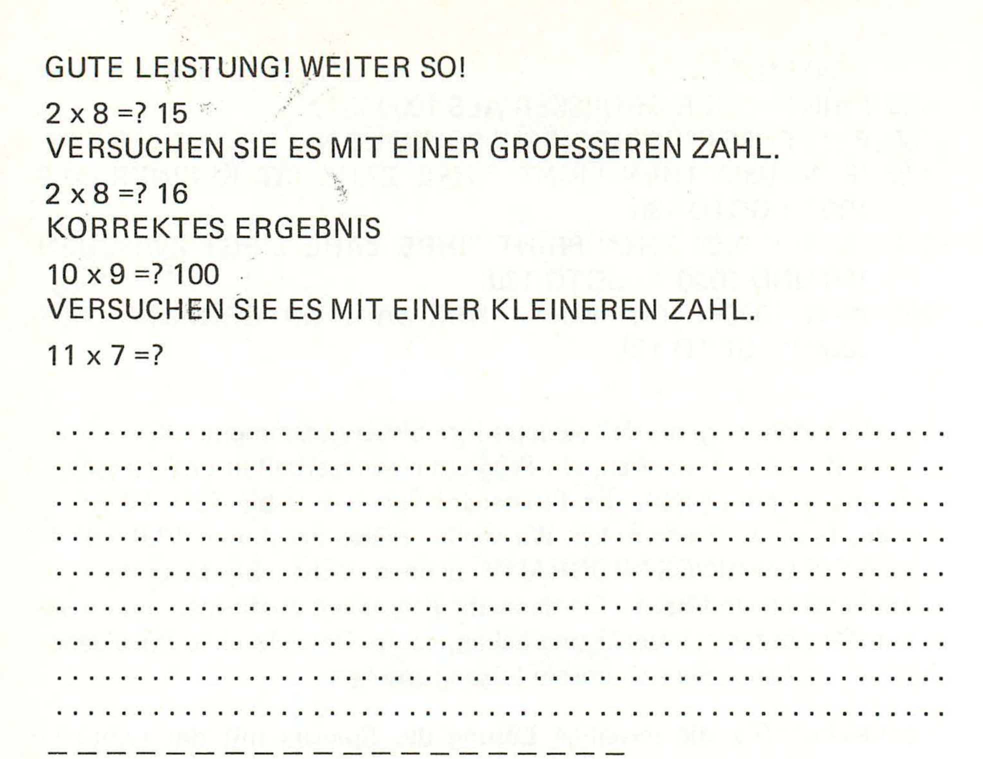
Liegt die gegebene Antwort über dem richtigen Ergebnis, drucken Sie VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL und wiederholen die Aufgabe.

Ist das Ergebnis richtig, teilen Sie dem Spieler mit, daß Sie oder er die richtige Zahl eingetippt hat.

Unser Programm verwendet drei verschiedene Mitteilungen, die bei einem korrekten Ergebnis ausgedruckt werden:

```
DAS IST RICHTIG!
KORREKTES ERGEBNIS
GUTE LEISTUNG! WEITER SO!
```

```
RUN
2 x 4 =? 8
```



GUTE LEISTUNG! WEITER SO!

2 x 8 =? 15
VERSUCHEN SIE ES MIT EINER GROESSEREN ZAHL.

2 x 8 =? 16
KORREKTES ERGEBNIS

10 x 9 =? 100
VERSUCHEN SIE ES MIT EINER KLEINEREN ZAHL.

11 x 7 =?

```
100 REM***MULTIPLIKATIONS-UEBUNGS-PROGRAMM
110 LET A=INT(12*RND(1))+1 : LET B=INT(12*RND(1))+1
120 PRINT : PRINT A; "X" ; B; "=" ; : INPUT C
130 IF C < A*B THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL." : GOTO 120
140 IF C > A*B THEN PRINT "VERSUCHEN SIE ES MIT EINER
    KLEINEREN ZAHL." : GOTO 120
150 LET R=INT(3*RND(1))+1 : ON R GOTO 160, 170, 180
160 PRINT "DAS IST RICHTIG!" : GOTO 110
170 PRINT "KORREKTES ERGEBNIS" : GOTO 110
180 PRINT "GUTE LEISTUNG! WEITER SO!" : GOTO 110
```

49. Wir erwähnten in Kapitel 2, daß einige der älteren BASIC-Versionen ein END-Statement als letztes Statement in einem Programm benötigen. Die meisten BASIC-Versionen für Heim-Computer erfordern dieses abschließende END nicht. Das END- und das verwandte STOP-Statement können jedoch sehr hilfreich ein, um ein Programm an einer anderen Stelle, als in der Zeile mit der höchsten Zeilennummer zu beenden.

Es ist oft sehr praktisch, wenn man ein Programm als Ergebnis einer IF-THEN-Entscheidung beenden oder anhalten kann. END oder STOP können beispielsweise ausgeführt werden, wenn die Bedingung WAHR ist, wie die Beispiele zeigen:

```
20 IF X = 10 THEN STOP
20 IF X = 10 THEN END
```

In ATARI-BASIC gibt der Computer, bei Verwendung von Stop, eine Mitteilung aus. Wie nachfolgende Zeile zeigt, enthält sie die Zeilennummer, in der das STOP-Statement auftrat und der Lauf endete.

```
STOPPED AT 20
■
```

Bei einem Lauf, der durch ein END-Statement beendet wird, wird nur das übliche READY angezeigt.

In der Zeile mit Mehrfach-Statements können END und STOP beispielsweise folgendermaßen verwendet werden:

```
120 IF Y*Q=100 THEN PRINT "DAS WAER'S, LEUTE" : END
120 IF Y*Q=100 THEN PRINT "DAS WAER'S, LEUTE" : STOP
```

Am Ende eines Mathematik-Übungsprogramms oder eines Spielprogramms können Sie beispielsweise folgende Lösung vorsehen:

```
310 PRINT "MOECHTEN SIE WEITERSPEILEN (JA ODER NEIN)"
    ;; INPUT A$
320 IF A$ = "NEIN" THEN END
330 GOTO 100
```

Fügen Sie in ein Zählschleifen-Programm eine Zeile ein, um dem Computer mitzuteilen, daß er das Zählen beenden soll, wenn F größer als 6 ist.

```
10 LET F = 1
20 ..........................
30 PRINT "F = "; F
40 LET F = F + 1
```

```
50 GOTO 20
---------------------
20 IF F > 6 THEN END
20 IF F > 6 THEN STOP
```

EIGENTEST

Arbeiten Sie jetzt bitte den anschließenden Eigentest durch. Sie können dabei feststellen, was Sie in Kapitel 4 gelernt haben.

1. Geben Sie die BASIC-Symbole für jeden der folgenden Vergleiche an:

....	gleich
....	kleiner als
....	größer als
....	kleiner oder gleich
....	größer oder gleich
....	ungleich

2. Beschreiben Sie jedes IF-THEN-Statement mit Worten. Das heißt, Sie sollen angeben, welche Anweisung das Statement dem Computer erteilt.
   a) IF G = X THEN 200 ....
   ....
   ....
   b) IF X > = 0 THEN C = C+1 ....
   ....
   ....
   c) IF N < > INT(N) THEN PRINT "N IST NICHT GANZAHLIG." : GOTO 210 ....
   ....
   ....
   d) IF A ∧ 2+B ∧ 2=C ∧ 2 THEN PRINT "JA, ES HANDELT SICH UM EIN RECHTWINKLIGES DREIECK." ....
   ....
   ....

3. Schreiben Sie für jede der folgenden Anweisungen ein IF-THEN-Statement.

   a) Wenn der Wert von N kleiner oder gleich 7 ist, gehe zu Zeile 15.

   ....

   b) Wenn A kleiner als 2*P ist, dann erhöhe den Wert von N um 1 und gehe zur Zeile 180. ....

   c) Wenn M/N gleich INT(M/N) ist, dann drucke die folgende Mitteilung aus: M IST GANZZAHLIG TEILBAR DURCH N. ....

   ....

4. Vervollständigen Sie das folgende Programm, um festzustellen, nach wievielen Jahren sich Ihr Sparguthaben verdoppelt hat. Verwenden Sie Mehrfach-Statements in Zeile 190.

```
100 REM***WIEVIELE JAHRE DAUERT ES, BIS SICH IHR GELD
    VERDOPPELT HAT?
110 PRINT "WENN SIE DIE HOEHE DES SPARGUTHABENS UND"
120 PRINT "DEN JAHRESZINSSATZ EINGEBEN, SAGE ICH"
130 PRINT "IHNEN, WIEVIELE JAHRE ES DAUERT, BIS SICH"
135 PRINT "IHR GELD VERDOPPELT HAT."
140 PRINT
150 PRINT "SPARGUTHABEN" ; : INPUT P
160 PRINT "ZINSSATZ" ; : INPUT R
170 LET N=1
180 LET A=P*(1+R/100) ↑ N
190 ..............................................
200 PRINT
210 PRINT "ANZAHL DER JAHRE:";N
220 PRINT "HOEHE DES SPARGUTHABENS ZU DIESEM
    ZEITPUNKT DM":A
RUN
WENN SIE DIE HOEHE DES SPARGUTHABENS UND
DEN JAHRESZINSSATZ EINGEBEN, SAGE ICH
IHNEN, WIEVIELE JAHRE ES DAUERT, BIS SICH
IHR GELD VERDOPPELT HAT.

SPARGUTHABEN? 1000
ZINSSATZ? 6

ANZAHL DER JAHRE: 12
HOEHE DES SPARGUTHABENS ZU SIESEM ZEITPUNKT:
DM 2012.2 (2012.19638)
```

5. Wie sieht der Lauf des folgenden Programms aus? Schreiben Sie ihn hin.

```
10 LET K = 1
20 PRINT K
30 LET K=K+1
40 IF K <=5 THEN 20
RUN
```

....

....

....

....

....

6. Wie wird ein Lauf des folgenden Programms aussehen?

```
10 LET K=1
20 IF K < 5 THEN K=K+1 : GOTO 20
30 PRINT K
RUN
```

....

....

7. Beschreiben Sie die Zufallszahlen, die durch jeden der folgenden Ausdrücke erzeugt werden.

   a) RND(1) ....

   ....

   b) 2*RND(1) ....

   ....

   c) INT(2*RND(1)) ....

   ....

   d) INT(2*RND(1))+1 ....

   ....

   e) INT(3*RND(1))–1 ....

   ....

   f) 3.14159*RND(1) ....

   ....

8. Hier sehen Sie ein noch unvollständiges Programm, welches das Werfen einer Münze ("Wappen oder Zahl?") simuliert. Bitte vervollständigen Sie es. (Hinweis: C kann den Wert 0 oder 1 als Zufallszahl

haben.)

```
100 REM***WAPPEN ODER ZAHL?
110 LET C = ..........................................
120 IF C=0 THEN PRINT "WAPPEN" : GOTO 110
130 IF C=1 THEN PRINT "ZAHL" : GOTO 110
RUN
WAPPEN
WAPPEN
ZAHL
WAPPEN
```

Antworten zum Eigentest

Die jeweils in Klammern stehenden Zahlen, geben in der schon bekannten Weise die Abschnitte in diesem Kapitel an, in denen der jeweilige Stoff behandelt wurde.

1. = gleich
   ⟨ kleiner als
   ⟩ größer als
   ⟨ = kleiner oder gleich
   ⟩ = größer oder gleich
   ⟨ ⟩ ungleich (Abschnitt 7)

2. a) Wenn der Wert von G gleich dem Wert von X ist, gehe zu Zeile 200. Anderenfalls (G=X ist FALSCH) fahre in der gewohnten Reihenfolge fort. (Abschnitte 1 – 8)

   b) Wenn der Wert von X größer oder gleich) ist, erhöhe den Wert von C um 1. Anderenfalls (X ⟩ = 0 ist FALSCH), führe den LET-Teil des Statements nicht aus. In beiden Fällen erfolgt die weitere Programmbearbeitung in der Reihenfolge der Zeilennummern. (Abschnitt 10)

   c) Wenn N ⟨ ⟩ INT(N) WAHR ist, bedeutet das, daß der Wert von N nicht ganzzahlig ist. Drucke in diesem Fall den String N IST NICHT GANZZAHLIG und gehe zur Zeile 210. Ist N jedoch eine ganze Zahl, wird N ⟨ ⟩ INT(N) FALSCH sein. In diesem Fall

werden weder PRINT noch GOTO ausgeführt, und der Computer arbeitet in der regulären Reihenfolge der Zeilennummern weiter. (Abschnitte 1 – 8, 20 – 23, 42)

d) Ist der Wert von A^2 plus B^2 gleich dem Wert von C^2, dann drucke den String JA, ES HANDELT SICH UM EIN RECHTWINKLIGES DREIECK. Anderenfalls (Bedingung ist FALSCH) drucke den String nicht. In beiden Fällen fahre jedoch mit der regulären Reihenfolge der Zeilennummern fort.

3. a) IF N <= 7 THEN GOTO 15
   b) IF A < 2*P THEN LET N=N+1 : GOTO 180
   c) IF M/N = INT(M/N) THEN PRINT "M IST GANZZAHLIG TEILBAR DURCH N."
   (Abschnitte 1 – 10)

4. 190 IF A < 2*P THEN LET N = N + : GOTO 180
   Solange die neu berechnete Höhe des Sparguthabens A kleiner als der doppelte Wert des Anfangs-Sparguthabens P ist, erhöhe das Jahr N um 1 und gehe zurück zur Zeile 180, um einen neuen Betrag zu errechnen. (Abschnitte 9 – 10)

5. RUN
   1
   2
   3
   4
   5
   Da sich das PRINT-Statement innerhalb der Schleife befindet, wird es bei jedem Durchlauf ausgeführt. (Abschnitt 8)

6. RUN
   5
   In diesem Beispiel befindet sich das PRINT-Statement außerhalb der Schleife. Es wird daher nur einmal ausgeführt, nachdem die Schleife vollendet wurde (Abschnitt 8)

7. a) Zahlen zwischen 0 und 1. Jede Zufallszahl ist größer als 0, aber kleiner als 1. Richtig wäre auch die folgende Antwort:
   0 < RND(1) < 1 (Abschnitte 16, 17)
   b) Zahlen zwischen 0 und 2. Jede Zufallszahl ist größer als 0, aber

kleiner als 2. Somit gilt:
0 < 2*RND(1) < 2 (Abschnitt 17 – 19)

c) 0 oder 1. Andere Werte sind nicht möglich. (Abschnitte 20, 23 – 26)

d) 1 oder 2. (Abschnitte 20, 23 – 26)

e) –1, 0 oder 1. (Abschnitte 20, 23 – 26)

f) Zahlen zwischen 0 und 3.14159. Jede Zufallszahl ist größer als 0, aber kleiner als 3.14159, so gilt:
0 < 3.14159*RND(1) < 3.14159
Da 3.14159 ist eine Näherung von π ist, könnten wir auch etwas unpräzise sagen, daß die Zufallszahlen zwischen 0 und π liegen. (Abschnitte 20, 23 – 26)

8. 110 LET C = INT(2*RND(1)) (Abschnitte 26, 27)

NOTIZEN

KAPITEL 5

READ und DATA arbeiten zusammen

Dieses Kapitel wurde so geschrieben, daß Sie Gelegenheit haben die bereits erlernten BASIC-Statements und Programmiertechniken zu üben und Ihrem Vorrat an Programmiertricks einige neue hinzufügen können. Sie werden anschließend ein größeres Verständnis für die Möglichkeiten des PRINT-Statements haben, um die Form der Ausgabe Ihres Programmes besser zu gestalten und wirkungsvollere Instruktionen zu schreiben. Zusätzlich werden Sie zwei häufig benutzte Statements erlernen, die immer zusammenarbeiten, um Variablen Werte zuzuordnen: die READ- und DATA-Statements, wenn Sie dieses Kapitel beendet haben, können Sie . . . .

- READ-Statements benutzen, um einer oder mehr Variablen gleichzeitig Werte oder Strings zuzuweisen, und zwar aus dem Vorrat im DATA-Statement;
- die Abstände in der Programmausgabe, durch die Verwendung von Komma und Semikolon zum Trennen der einzelnen Ausgaben innerhalb des PRINT-Statements beeinflussen;
- die Standard-Druck-Position im Ausdruck indentifizieren und PRINT-Statements unter Verwendung trennender Kommas schreiben.

1. Sie haben gesehen, wie LET- und INPUT-Statements dazu verwendet werden können, Variablen Werte zuzuordnen. Eine dritte Methode verwendet eine Kombination von zwei Statements, nämlich READ und DATA, um Variablen Werte zuzuweisen.

```
10 READ X
20 PRINT "BEI DIESEM SCHLEIFENDURCHLAUF IST X = " ; X
30 GOTO 10
40 DATA 10, 15, 7, 3.25, 11
RUN
BEI DIESEM SCHLEIFENDURCHLAUF IST X = 10
BEI DIESEM SCHLEIFENDURCHLAUF IST X = 15
```

```
BEI DIESEM SCHLEIFENDURCHLAUF IST X = 7
BEI DIESEM SCHLEIFENDURCHLAUF IST X = 3.25
BEI DIESEM SCHLEIFENDURCHLAUF IST X = 11
ERROR– 6 AT LINE 10
```

Das Statement 10 READ X befiehlt dem Computer, einen Wert aus dem DATA-Statement zu lesen und diesen Wert der Variablen X zuzuweisen. Jedesmal bei der Ausführung des READ-Statements (bei jedem Schleifendurchlauf also), liest der Computer den nächsten Wert aus dem DATA-Statement und weist ihn der Variablen X zu. Der Computer "merkt" sich dabei, welche Werte der Reihe nach gelesen wurden. Dies geschieht, indem er gleichsam einen "Zeiger" im Data-Statement bei jedem Lesevorgangg um einen Schritt weiterrückt.

Wieviele Werte enthält das obige DATA-Statement? . . . .

– –

5

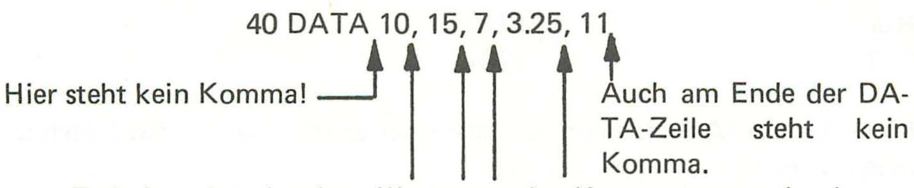
2. Als der Computer das Programm ausführte, las er bei jedem Schleifendurchlauf einen Wert aus dem DATA-Statement und druckte ihn aus. Beim letzten Durchlauf schließlich versuchte er noch keine weitere Zahl zu finden. Da sich im DATA-Statement jedoch keine mehr befand, durckte er ERROR 6 IN LINE 10 aus, was bedeutet: "In Zeile 10 sind keine weiteren Daten mehr vorhanden." Es handelt sich dabei eigentlich garnicht um einen Fehler. Diese Angabe informiert Sie lediglich darüber, daß der Computer alle zur Verfügung stehenden Daten verwendet hat und versuchte, weitere zu finden, was aber nicht möglich war.

Wieviele neue Werte wurden der READ-Variablen beim Lauf des Programms in Abschnitt 1 zugeordnet? . . . .

– –

5 Werte wurden zugeordnet

3. Sehen Sie sich einmal das Format des DATA-Statements näher an:



DATA-Statements können ganze Zahlen, Zahlen mit Dezimalbruchanteil, Zahlen in Gleitkomma-Darstellung oder negative Zahlen enthalten.

DATA-Statements dürfen keine Variablen, arithmetische Operationen, andere Funktionen oder Brüche enthalten.

Zulässig ist:	90 DATA 3, 8, 2.5
Nicht zulässig ist:	95 DATA 2+3, 1/4, 2/4, 2/4, 7*8

Schreiben Sie ein DATA-Statement für die folgenden Werte:

342
1256
205
60.25
–412
2.05E8

. . . .

– –

60 DATA 342, 1256, 205, 60.25, –412, 2.05E8

Ihre Zeilennummer kann natürlich anders lauten. Denken Sie daran, daß bei großen Zahlen, wie beispielsweise 1256 im obigen Beispiel, keine Kommas benutzt werden dürfen! Gleitkommadarstellung ist jedoch zulässig.

4. DATA-Statements können an beliebiger Stelle im Programm angeordnet werden:

10 READ N	10 DATA 123
20 PRINT N	20 READ N
30 DATA 123	30 PRINT N

```
RUN                          RUN
123                          123
```

Kann das DATA-Statement so angeordnet werden, wie es das folgende Beispiel zeigt? . . . .

```
10 READ N
20 DATA 123
30 PRINT N
```

– –

Ja

5. Die folgenden DATA-Statements sind nicht in korrektem BASIC geschrieben. Bitte geben Sie jeweils an, was falsch ist.

a) 20 DATA, 15, 32, 85, 66 . . . .
b) 30 DATA 22.3; 81.1; 66.66; 43.22 . . . .

– –

a) Nach DATA darf kein Komma stehen!
b) Im DATA-Statement müssen Kommas statt Semikolon verwendet werden.

6. Das folgende Programm wandelt Zoll in Zentimeter um. Ein Zoll sind 2.54 cm. Wenn das Programm läuft, druckt der Computer den Wert in Zoll in eine Zeile und die entsprechende Angabe in Zentimetern in die nächste. Tragen Sie bitte in der leeren Zeile das fehlende Statement ein.

```
10 REM***UMWANDLUNG VON ZOLL IN ZENTIMETER
20 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
30 PRINT
40 PRINT "ZOLL =";Z
50 PRINT "ZENTIMETER =";2.54*Z
60 GOTO 20
90 DATA 1, 8

RUN
ZOLL = 1
ZENTIMETER = 2.54
```

```
ZOLL = 8
ZENTIMETER = 20.32
ERROR– 6 AT LINE 20
```

– –

```
20 READ Z
```

7. Schreiben Sie jetzt selbst ein Programm zur Umwandlung von Unzen in Gramm. Eine Unze sind 28.35 g, abgerundet auf zwei Dezimalstellen. Verwenden Sie ein DATA-Statement zur Aufnehme der Angaben in Unzen, die Sie in Gramm umgerechnet haben möchten. Hier sehen Sie, wie das Programm ablaufen sollte.

```
RUN

UNZEN = 1
GRAMM = 28.35

UNZEN = 13
GRAMM = 368.55
ERROR– 6 AT LINE 20
```

....

....

....

....

....

....

....

– –

```
10 REM***UMWANDLUNG VON UNZEN IN GRAMM
20 READ U
30 PRINT
40 PRINT "UNZEN = " ; U
50 PRINT "GRAMM = " : 28.35*U
60 GOTO 20
90 DATA 1, 13, 16
```

8. Schreiben Sie das Programm "ICH BIN DIE TEUERSTE ADDIER–MASCHINE DER WELT" aus Kapitel 3, Abschnitt 45 unter Verwendung von READ- und DATA-Statements statt eines INPUT-State-

ments so um, daß ein Lauf des Programms wie nachgolgend abgebildet aussieht. Sehen Sie sich den Lauf genau an, um die einzelnen Werte im DATA-Statement zu ermitteln.

```
RUN
X = 12
SUMME BIS JETZT 12
X = 43
SUMME BIS JETZT 55
X = 33
SUMME BIS JETZT 88
X = 92
SUMME BIS JETZT 180
X = 76.25
SUMME BIS JETZT 256.25
ERROR- 6 AT LINE 120
```

....
....
....
....
....
....
....
....

```
100 REM***DIE TEUERSTE ADDIERMASCHINE DER WELT
110 LET T=0
120 READ X
130 LET T=T+X
140 PRINT "X = " ; X
150 PRINT "SUMME BIS JETZT" ; T
160 PRINT
170 GOTO 120
180 DATA 12, 43, 33, 92, 76.25
```

Ihre Zeilennummern können natürlich anders lauten.

9. Erinnern Sie sich noch an unser Programm zur Berechnung des Mittelwertes oder Durchschnitts einer Gruppe von Zahlen? Wenn nicht, sehen Sie sich noch einmal Abschnitt 12 in Kapitel 4 an. Nachfolgend finden Sie das Programm in einer umgeschriebenen Version, und zwar unter Verwendung von READ- und DATA-Statements. Wir haben außerdem mehrere Anmerkungszeilen (REM) vorgesehen, um zu erklären, was jeweils geschieht.

```
100 REM***EIN FREUNDLICHES MITTELWERTPROGRAMM
110 REM***INITIALISIERE T (FUER DIE SUMME) UND N (FUER
    DIE ANZAHL)
120 T=0 : N=0
130 REM***LESEN EINER ZAHL X. WENN ES SICH DABEI UM
140 REM***DAS FLAG 1E97 HANDELT, SOLL DER DRUCK
145 REM***ERFOLGEN. ANDERNFALLS SOLLTEN T UND X
    ERNEUERT WERDEN.
150 READ X : IF X=1E97 THEN 180
160 T=T+X : N=N+1 : GOTO 150
170 REM***DRUCKE DIE ERGEBNISSE
180 PRINT "N = " ; T
200 PRINT "MITTELWERT = " ; T/N
900 REM***DATEN FOLGEN
910 DATA 10,3,–9,–15,–23,–25,–30,1E97 ◄———— Das Flag!

RUN
N = 7
SUMME = –89
MITTELWERT = –12.71428571
```

Um dieses Programm mit einem anderen Datensatz laufen zu lassen, ersetzen Sie einfach Zeile 910 durch ein oder mehr DATA-Statements, welche die neuen Daten und das Flag 1E97 enthalten. Sie erinnern sich sicher noch daran, daß Sie nur die neuen Statements mit den alten Zeilennummern zu schreiben brauchen, wenn Sie ein Statement ersetzen möchten. Nehmen Sie an, wir wollen beispielsweise die folgenden Daten verwenden: 63, 72, 50, 55, 75, 67, 59, 61 und 64. Schreiben Sie das zugehörige DATA-Statement.

910 DATA ....

– –

910 DATA 63, 72, 50, 55, 75, 67, 59, 61, 64, 1E97 (Haben Sie auch das Flag nicht vergessen?)

10. Das folgende Programm veranlaßt den Computer, die Zahlen aus einem DATA-Statement zu lesen, jedoch nur diejenigen auszudrucken, die größer als Null, also positiv sind. Zahlen die kleiner oder gleich Null sind werden nicht gedruckt.

```
10 READ X
20 IF X > 0 THEN PRINT "X = " ; X
30 GOTO 10
40 DATA 3, 7, 0, -2, 5, -1, 7, 8, 0, -3
RUN
X = 3
X = 7
X = 5
X = 7
X = 8
ERROR- 6 AT LINE 10
```

Anmerkung:

Die Zahl 7 wird zweimal ausgedruckt, da sie auch zweimal im DATA-Statement vorkommt.

Sehen Sie sich die Zahlen im DATA-Statement bitte genau an und beantworten Sie folgende Fragen:

a) Für welche Zahlen ist die Bedingung X > 0 WAHR? . . . .
b) Wozu veranlaßt Zeile 20 den Computer, wenn X > 0 WAHR ist?
. . . .
c) Für welche Zahlen ist die Bedingung X > 0 FALSCH? . . . .
d) Was geschieht in Zeile 20, wenn X > 0 FALSCH ist? . . . .

— —

a) 3, 7, 5, 7, 8
b) die Mitteilung X= gefolgt vom Wert von X ausdrucken
c) 0, –2, –1, 0, –3
d) Nichts. Der PRINT-Teil von Zeile 20 wird nicht ausgeführt und der Computer geht weiter zur Zeile 30.

11. Wie wird ein Lauf dieses Programms aussehen?

```
10 READ X
20 IF X < 0 THEN PRINT "X = " ; X
30 GOTO 10
40 DATA 3, 7, 0, –2, 5, –1, 7, 8, 0, –3
RUN
```

....

....

....

– –

```
RUN
X = –2
X = –1
X = –3
ERROR– 6 AT LINE 10
```

← Haben Sie daran gedacht?

12. Vervollständigen Sie bitte das nachfolgende Programm so, daß sich der angegebene Lauf ergibt.

```
10 READ X
20 ..................................................
30 GOTO 10
40 DATA 3, 7, 0, –2, 5, –1, 7, 8, 0, –3
RUN
X = 0
X = 0
ERROR– 6 AT LINE 10
```

Es werden zwei Nullen ausgedruckt, da sich im DATA-Statement zwei Nullen befinden.

– –

```
20 IF X=0 THEN PRINT "X = "; X
```

13. Bitte vervollständigen Sie zur Übung die nachfolgenden Statements. Falls möglich, probieren Sie jedes Statement auf Ihrem Computer aus.

a) Vervollständigen Sie die Zeile 20 so, daß nur von Null verschiedene

Zahlen gedruckt werden. 20 IF . . . . . THEN PRINT "X = " ; X

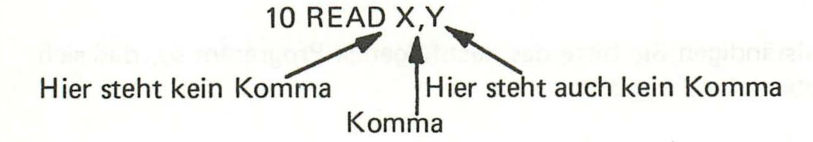
b) Vervollständigen Sie Zeile 20 so, daß der Computer nur Zahlen ausdruckt, die größer oder gleich Null sind.
20 IF . . . . THEN PRINT "X = " ; X

c) Vervollständigen Sie Zeile 20 so, daß der Computer Zahlen ausdruckt, die kleiner oder gleich 3 sind. 20 IF . . THEN PRINT "X = " ; X

– –

a) X <> 0 (Der Computer druckt: 3, 7, –2, 5, –1, 7, 8 und –3.)
b) X >= 0 (Der Computer druckt: 3, 7, 0, 5, 7, 8 und 0.)
c) X <= 3 (Der Computer druckt: 3, 0, –2, –1,) und –3.)

14. In BASIC dürfen auf eine READ-Instruktion mehr als eine Variable folgen. Nachfolgend finden Sie ein Beispiel. Verwenden Sie Kommas, um die Variablen zu trennen.



Dieses Statement veranlaßt den Computer, zwei Werte aus dem DATA-Statement zu entnehmen und sie in der entsprechenden Reihenfolge den beiden READ-Variablen zuzuordnen.

Vervollständigen Sie bitte den Lauf des folgenden Programms:

```
10 READ X,Y
20 PRINT X,Y
30 GOTO 10
40 DATA 10, 20, 30, 40
RUN
10            20
. . . . . . .   . . . . . . . . . .
ERROR– 6 AT LINE 10
```

– –

30 40

Beim zweiten Schleifendurchlauf wurden X und Y diese Werte zugewiesen und in Zeile 20 ausgedruckt. (Die Abstände zwischen den einzelnen Zahlen werden wir später besprechen).

15. Füllen Sie bitte die freien Stellen im Programm und im Lauf des Programms aus.

```
10 ..............................................
20 PRINT A + B
30 GOTO 10
40 DATA 3, 5, 6, 4, 7, 9
RUN
8
...................
...................
ERROR– 6 AT LINE 10
```

```
10 READ A,B
RUN
8
10
16
```

16. Sie können in einem Programm mehr als ein READ-Statement vorsehen. Alle READ-Statements weisen ihren Variablen die Werte jedoch aus dem gleichen DATA-Statement zu. Ein Wert aus einem DATA-Statement wird einer READ-Variablen in der Reihenfolge zugeordnet, in der der Computer beim Ablauf des Programms auf ein READ-Statement trifft.

```
10 READ P
20 READ Q
30 PRINT P
40 PRINT Q
50 GOTO 10
60 DATA 3, 5, 6, 4, 7, 9
```

```
RUN
3 ⎤
  ⎬——Der erste Wert wird P, der zweite Q zugeordnet.
5 ⎦
6 ⎤
  ⎬——Zweiter Schleifendurchlauf: der dritte Wert wird
4 ⎦    P, der vierte Q zugewiesen.
7
9
ERROR– 6 AT LINE
```

Welcher Wert wird P im dritten Schleifendurchlauf zugewiesen? ......
Welcher Wert wird Q zugewiesen? ....

— — — — — — — — — — — — — — — — — — —

7, 9

17. Schreiben Sie ein Programm, das drei READ-Statements verwendet, um drei verschiedenen READ-Variablen X, Y und Z Werte zuweist; dann die Summe von X, Y und Z ausdruckt; dann zurückspringt und den Prozeß wiederholt, bis alle Daten verbraucht sind.

Schreiben Sie auf, was der Computer ausdruckt, wenn Ihr Programm läuft. Hier sind die Daten für das DATA-Statement: 3, 5, 6, 4, 7, 9, 2, 5, 2.

....
....
....
....
....
....

— — — — — — — — — — — — — — — — — — —

```
10 READ X
20 READ Y
30 READ Z
40 PRINT X + Y + Z
50 GOTO 10
60 DATA 3, 5, 6, 4, 7, 9, 2, 5, 2
RUN
14
20
```

```
9
ERROR– 6 AT LINE 10
```

18. Die folgende Frage haben wir 50 Personen gestellt:

VERSTEHT SIE IHR COMPUTER?
1. JA
2. NEIN

Jede der 50 Antworten lautete entweder 1 (JA) oder 2 (NEIN). Die Antworten sind in den nachfolgenden 5 DATA-Statements zu sehen. Auf die letzte Antwort folgt –1. Dieses Flag signalisiert das Ende der Daten.

```
900 REM***DATEN* 1 = JA, 2 = NEIN, –1 = ENDE DER DATEN
910 DATA 1,2,2,2,1,2,1,2,1,2
920 DATA 2,1,1,1,2,1,2,2,2,1
930 DATA 2,2,2,1,2,1,2,2,1,2
940 DATA 1,1,1,1,2,1,2,2,1,1
950 DATA 2,2,2,2,1,1,1,2,1,2,–1
```

Wieviele JA-Antworten wurden gegeben? ....
Wieviele NEIN-Antworten wurden gegeben? ....

Schreiben Sie die Anzahl der Antworten, die "JA" lauteten in das mit "J" gekennzeichnete Feld, die Anzahl der "NEINs" in das mit "N" gekennzeichnete Feld.

J		N

– –

J = 23;   N = 27

19. Hier sehen Sie ein Programm, das die Antworten aus den DATA-Statements liest und die Anzahl der "JA" und "NEIN" zählt. Die Variable J wird zum Zählen der "JA"-Stimmen, die Variable N zum Zahlen der "NEIN"-Stimmen verwendet.

```
100 REM***BEFRAGUNGS-ANALYSEPROGRAMM
110 REM***INITIALISIERUNG: SETZE ALLE ZAEHLVARIABLEN
```

```
    AUF NULL
120 J=0 : N=0
200 REM***LIES UND ZAEHLE DIE ANTWORTEN
210 READ A : IF A = – 1 THEN 410
220 IF A=1 THEN LET J=J+1 : GOTO 210
230 IF A=2 THEN LET N=N+1 : GOTO 210
400 REM***DRUCKE DIE ERGEBNISSE AUS
410 PRINT
420 PRINT "JA:" ; J
430 PRINT "NEIN:" ; N
900 REM***DATA*  1=JA,  2=NEIN,  –1=ENDE  DER  DATEN
910 DATA 1,2,2,2,1,2,1,2,1,2
920 DATA 2,1,1,1,2,1,2,2,2,1
930 DATA 2,2,2,1,2,1,2,2,1,2
940 DATA 1,1,1,1,2,1,2,2,1,1
950 DATA 2,2,2,2,1,1,1,2,1,2,–1

RUN
JA: 23
NEIN: 27
```

a) Welcher Abschnitt des Programms bildet eine Schleife, die für jeden Wert in den DATA-Statements durchlaufen wird? Zeilen . . . bis . . .
b) Welches Statement liest einen Wert und legt ihn in "Schachtel" A? . . . .
c) Welche Zeile entscheidet, ob eine Antwort "JA" lautet und erhöht den Stand der Ja-Stimmen um 1, wenn es der Fall ist? . . . .
d) Welche Zeile entscheidet, ob eine Antwort "NEIN" lautet und erhöht, wenn es der Fall ist, den Stand der "Nein"-Stimmen um 1?
. . . .

– –

a) Zeilen 210 bis 230 b) Zeile 210
c) Zeile 220 d) Zeile 230

20. Sehen Sie sich jetzt die folgende Befragung an:

VERSTEHT SIE IHR COMPUTER?
1. JA
2. NEIN
3. MANCHMAL

Modifizieren Sie das Programm in Abschnitt 19 so, daß der Computer die Ja-, Nein- und Manchmal-Stimmen zählt. Verwenden Sie die Variable J um die Ja-Stimmen zu zählen. Für die Nein-Stimmen verwenden Sie bitte N und für die Manchmal-Stimmen den Buchstaben M. Benutzt werden sollen die folgenden Daten.

2,1,3,2,3,3,1,3,3,2,1,2,1,2,1,1,3,3,–1

Unter Verwendung dieser Daten sollen die folgenden Ergebnisse beim Lauf des Programms ausgedruckt werden:

JA: 6
NEIN: 5
MANCHMAL: 7

....
....
....
....
....

– –

Hier sind unsere Änderungen:

```
120 J=0 : N=0 : M=0
240 IF A=3 THEN LET M=M+1 : GOTO 210
440 PRINT "MANCHMAL:"; M
900 REM***DATA: 1=JA, 2=NEIN, 3=MANCHMAL, –1=ENDE
    DER DATEN
910 DATA 2,1,3,2,3,3,1,3,3,2,1,2,1,2,1,1,3,3,–1
920
930
940
950
```

Wir haben die Zeilen 920, 930, 940 und 950 aus dem Programm entfernt. Wenn sich das Programm im Computer befindet, läßt sich dies sehr einfach durchführen, indem man die Zeilennummer schreibt und auf RETURN drückt.

21. In Kapitel 2 verwendeten wir PRINT-Statements in der folgenden Form:

PRINT e

worin e ein numerischer Ausdruck ist.

Beispielsweise:

PRINT 7 + 5

numerischer Ausdruck

Ein PRINT-Statement dieser Form fordert den Computer auf, den numerischen Ausdruck zu berechnen und das Ergebnis auszudrucken.

Das folgende PRINT-Statement weist den Computer an, gleich vier numerische Ausdrücke zu berechnen und die vier Ergebnisse auszudrucken.

Wir schreiben: PRINT 7+5, 7–5, 7*5, 7/5

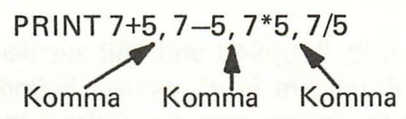
Der Computer schreibt: 12 2 35 1.4

Zeichnen Sie im obigen Beispiel Pfeile ein, die jeden numerischen Ausdruck mit seinem berechneten und ausgedruckten Wert verbinden. Wir haben bereits den ersten Pfeil eingezeichnet, der den Ausdruck 7+5 mit dem Ergebnis "12" verbindet.

PRINT 7+5, 7–5, 7*5, 7/5

12 2 35 1.4

22. Wenn ein PRINT-Statement mehr als einen Ausdruck enthält, werden die einzelnen Ausdrücke durch Kommas getrennt.



Das folgende PRINT-Statement weist den Computer an, die Werte von 2+3, 2–2, 2*3 und 2/3 zu berechnen und auszudrucken. Wir haben jedoch vergessen, Kommas vorzusehen. Bitte fügen Sie sie in der für BASIC korrekten Weise ein.

PRINT 2+3 2–2 2*3 2/3

Print 2+3, 2–3, 2*3, 2/3

Anmerkung:

Auf PRINT und 2/3 folgen keine Kommas. Ein Komma hinter PRINT verursacht eine Fehlermeldung. Ein Komma hinter 2/3, am Ende des PRINT-Statements hat einen speziellen Zweck, den wir später noch kennenlernen werden.

23. Füllen Sie bitte in den folgenden Zeilen die freien Stellen aus. (Denken Sie daran, daß Sie bei direkten Statements keine Zeilennummern benötigen und nur auf RETURN drücken müssen, um den Computer zu veranlassen, das Statement auszuführen.)

Wir schreiben:	PRINT 3*3, 5*5, 7*7
Der Computer schreibt:	....
Wir schreiben:	PRINT 2*3+4*5, (2+3)*(4+5)
Der Computer schreibt:	....

9	25	49
26	45	

24. Jetzt sind Sie an der Reihe. Wir wollen uns noch einmal mit den bereits bekannten drei Fahrrädern mit Reifendurchmessern von 20, 24 und 26 Zoll beschäftigen. Wir wollten seinerzeit feststellen, wie groß der Umfang jedes Reifens ist, oder anders gesagt, welche Strecke das Rad mit einer Umdrehung zurücklegt. Dazu müssen die folgenden drei Ausdrücke berechnet werden:

20-Zoll-Reifen	3.14*20	(erster Ausdruck)
24-Zoll-Reifen	3.14*24	(zweiter Ausdruck)
26-Zoll-Reifen	3.14*26	(dritter Ausdruck)

Schreiben Sie ein PRINT-Statement, das den Computer anweist, die drei Ausdrücke zu berechnen und die Ergebnisse auszudrucken.

Sie schreiben: ....

Der Computer gibt aus: 62.8 75.36 81.64

PRINT 3.14*20, 3.14*24, 3.14*26

25. In ATARI-BASIC sind vier Standard-PRINT-Positionen vorgesehen. Ein Komma in einem PRINT-Statement veranlaßt den Cursor, zur nächsten verfügbaren Standard-Position weiterzugehen. Sehen Sie sich das Beispiel an und füllen Sie die leeren Stellen aus.

Wir schreiben: PRINT 1, 2, 3, 4

Der Computer: 1 2 3 4

Positon 1 Position 2 Position . . Position . .

3; 4

26. Positive Zahlen und Null werden ohne Vorzeichen ausgedruckt. Bitte sehen Sie sich an, was geschieht, wenn negative Zahlen gedruckt werden sollen.

Wir schreiben: PRINT –1, –2, –3, –4

Der Computer : –1 –2 –3 –4

Position 1 Position 2 Postition 3 Position 4

Wie unterscheidet sich der Ausdruck negativer Zahlen von den positiven Zahlen? ....

Negative Zahlen werden mit einem Minuszeichen (–), gefolgt von den einzelnen Ziffern der Zahl ausgedruckt.

27. Was geschieht, wenn in einem PRINT-Statement mehr als vier auszudruckenden Zahlen oder Zeichen enthalten sind? Sehen Sie sich am folgenden Beispiel an, wie der Computer in diesem Fall vorgeht.

Wir schreiben: PRINT 1, 2, 3, 4, 5, 6, 7

Der Computer: 1 2 3 4
5 6 7

Beschreiben Sie, was geschehen ist: . . . .

. . . .

— —

Der Computer druckte die 7 Zahlen in zwei Zeilen, und zwar vier Zahlen in die erste, die restlichen drei Zahlen in die zweite Zeile.

28. Bitte Schreiben Sie auf, wie der Computer, gemäß den Regeln für Kommas innerhalb von PRINT-Statements, die folgende Anweisung ausführen wird.

Wir schreiben: PRINT 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Der Computer: . . . .

. . . .

. . . .

— —

```
1        2        3        4
5        6        7        8
9        10
```

29. Statt Kommas können wir zwischen zwei Zahlen auch jeweils ein Semikolon setzen. Sehen Sie sich einmal an, was dann geschieht.

Wir schreiben: PRINT –1;–2;–3;–4;–5;–6;–7
Der Computer: –1–2–3–4–5–6–7

Wir schreiben: PRINT1;2;3;4;5
Der Computer: 12345

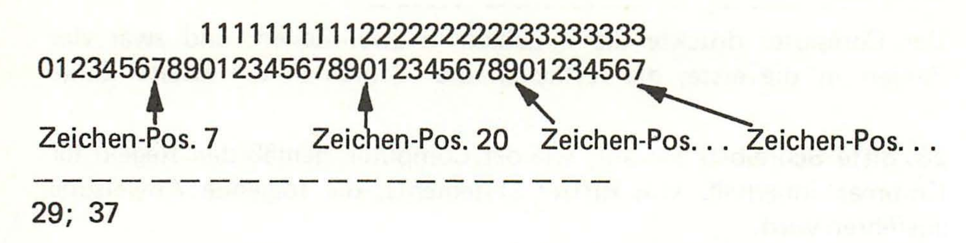
Die Verwendung von Semikolons führt also dazu, daß die einzelenen Ziffern direkt, ohne Zwischenräume, aneiandergehängt werden. Kommas dagegen weisen den Computer an, die vorbestimmten Druck-Positionen zu verwenden. Um also beispielsweise Ergebnisse dicht aneinander auszudrucken, müssen Sie . . . . verwenden.

— —

Semikolons

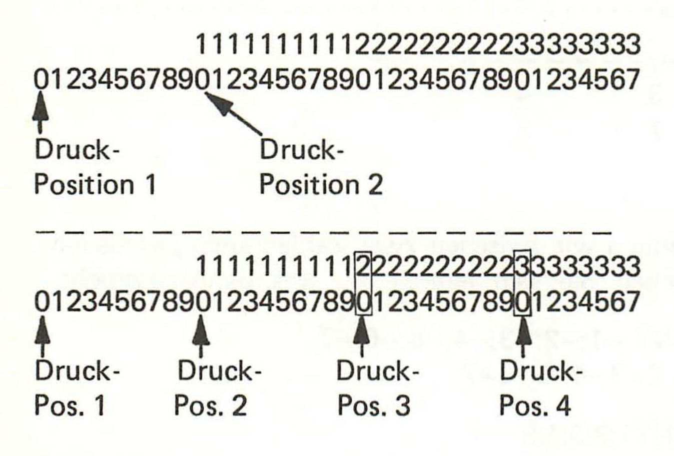
30. Der Bildschirm kann in jeder Zeile 38 Zeichen aufnehmen. Das heißt, er verfügt über 38 Zeichen-Positionen, die von 0 am linken bis

37 am rechten Rand durchnumeriert sind. Vervollständigen Sie bitte das folgende Diagramm der Zeichenpositionen auf dem Bildschirm.



29; 37

31. Die vier Standard-Druck-Positionen beginnen bei der Zeichen-Position 0. Wir haben die Druckpositionen 1 und 2 markiert. Kreisen Sie bitte im folgenden Diagramm die Positionen 3 und 4 ein.



Anmerkung:

Eine Standard-Druck-Position umfaßt 10 Zeichen-Positionen. Beispielsweise belegt die Druck-Position 1 die Zeichen-Positionen 0 bis 9, insgesamt also 10 Plätze. Die Druck-Position 2 belegt die Zeichen-Positionen 10 bis 19 usw.

32. Sehen Sie sich jetzt an, was geschieht, wenn wir das Komma am Ende eines PRINT-Statements hinschreiben.

```
10 LET N=1
20 PRINT N,  <——— Komma
30 LET N=N+1
```

```
40 GOTO 20
RUN
1          2          3          4
5          6          7          8
9          10         11
STOPPED AT LINE 20
```

An dieser Stelle haben wir auf die Taste BREAK gedrückt, um den Lauf zu beenden.

Bitte erinnern Sie sich daran, was das Komma bewirkt: es weist den Computer an, zur nächsten Standard-Druck-Position weiterzugehen. Er springt daher, nach dem Drucken der Zahl 1, zur Druckposition 2; nach dem Ausdrucken der Zahl 2 geht er weiter zur Druck-Position 3 usw. Nachdem auch die Position 4 belegt wurde, springt er zur nächsten Zeile und druckt dort weiter.

Geben Sie für die ersten 7 Zahlen des folgenden Programms an, wohin sie gedruckt werden.

```
10 LET N=1
20 PRINT N,
30 LET N=N+2
40 GOTO 20
RUN
----------------------
1          3          5          7
9          11         13
```

33. Was wird Ihrer Ansicht nach geschehen, wenn wir am Ende eines PRINT-Statements ein Semikolon vorsehen? Probieren wir es doch einfach einmal aus!

```
10 LET N=1
20 PRINT N;
30 LET N=N+1
40 GOTO 20
```

```
RUN
12345678901112131415161718 19
STOPPED AT LINE 20
```

Hier haben wir die Taste BREAK gedruckt

Schreiben Sie jetzt Zeile 20 so um, das der Computer hinter jeder gedruckten Zahl einen Zwischenraum vorsieht.

20 PRINT ....

_ _

20 PRINT N; " "; (Sehen Sie zwischen den Anführungszeichen eine Leerstelle vor.)

Wenn Sie vor jeder Zahl eine Leerstelle wünschen, verfahren Sie bitte folgendermaßen:

20 PRINT " ";N;

Die Leerstelle innerhalb der Anführungszeichen steht jetzt vor der Zahl (N).

34. Bisher haben wir uns nur damit befaßt, was geschieht, wenn Zahlen innerhalb eines PRINT-Statements durch Kommas oder Semikolons getrennt werden.
Jetzt wollen wir es einmal mit Strings versuchen.

PRINT "A", "B", "C", "D"

A	B	C	D
↑	↑	↑	↑
Druck-Pos. 1	Druck-Pos. 2	Druck-Pos. 3	Druck-Pos. 4

Jetzt sind Sie an der Reihe. Vervollständigen Sie die Ausgabe für das folgende Statement, so wie es der Computer tun würde.

PRINT "A", "B", "C", "D", "E", "F", "G"

....

....

_ _

```
A         B         C         D
E         F         G
```

Wie bei Zahlen veranlaßt ein Komma auch bei Strings den Computer dazu, zur jeweils nächsten Standard-Druck-Position vorzurücken.

35. Die Verwendung von Kommas zur Vorgabe der Druck-Position ist beispielsweise zum Drucken von Überschriften sehr nützlich. Das soll am Beispiel des abgeänderten Programms zum Umrechnen von Zoll in Zentimeter aus Abschnitt 6 demonstriert werden.

```
10 REM***UMRECHNUNGSTABELLE ZOLL IN ZENTIMETER
20 PRINT "ZOLL","ZENTIMETER"
30 READ I
40 PRINT I,2.54*I
50 GOTO 30
90 DATA 1,8,12
RUN
ZOLL          ZENTIMETER
1             2.54
8             20.32
12            30.48
ERROR- 6 IN 30
```

Beachten Sie die Verwendung der Kommas in den Zeilen 20 und 40. Die vom Statement in Zeile 40 ausgedruckten Werte werden dadurch exakt unter den von Zeile 20 gedruckten Spaltenüberschriften angeordnet.

Jetzt sind Sie an der Reihe. Ändern Sie das Programm zur Umwandlung von Unzen in Gramm (Abschnitt 7) so ab, daß der Lauf aussieht, wie nachfolgend abgebildet.

```
RUN
UNZEN         GRAMM
1             28.35
13            368.55
16            453.6
ERROR- 6 AT LINE 30
```

....

....

....

....

....

....

_ _

```
10 REM***UMWANDLUNGSTABELLE UNZEN IN GRAMM
20 PRINT "UNZEN", "GRAMM"
30 READ Z
40 PRINT Z,28.35*Z
50 GOTO 30
90 DATA 1,13,16
```

38. Sehen wir uns jetzt einmal an, was geschieht, wenn wir in einem String Semikolons verwenden.

```
PRINT "A"; "B"; "C"; "D"; "E"
ABCDE
```

Das Semikolon führt dazu, daß alle Strings, ohne Zwischenraum, aneinander gedruckt werden.
Geben Sie an, was geschieht, wenn der Computer das folgende PRINT-Statement ausführt.
PRINT "DAS"; "IST"; "COMPUTER"; "PROGRAMMIERUNG?"

....

_ _

DASISTCOMPUTERPROGRAMMIERUNG?

39. Die Ausgabe in Abschnitt 38 ist sehr unübersichtlich und schwer zu lesen. wenn Sie zwischen den einzelenen Strings Leerschritte haben möchten, müssen Sie sie an den gewünschten Stellen vorsehen. Z.B:
PRINT "DAS ";"IST ";"COMPUTER ";"PROGRAMMIERUNG?"

Wie wird der Computer diese Zeile ausdrucken?

....

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

DAS IST COMPUTER PROGRAMMIERUNG?

40. Jetzt wollen wir einmal ein Programm ausprobieren, das den Ausdruck von Strings bewirkt, die String-Variablen zugewiesen wurden.

```
 5 DIM A$(20),B$(20),C$(20)
10 LET A$="COMPUTER "
20 LET B$="SIND"
30 LET C$=" INTERESSANT."
40 PRINT A$;B$;C$
50 PRINT A$,B$,C$
```

In Zeile 5 wird in der uns schon bekannten Weise Speicherplatz für die drei Strings reserviert. Achten Sie bitte auch auf die Leerschritte innerhalb der Anführungszeichen in den Zeilen 10 und 30.

Schreiben Sie jetzt auf, was der Computer drucken wird, wenn das obige Programm läuft:

RUN

. . . .

. . . .

– – – – – – – – – – – – – – – – – –

```
RUN
COMPUTER SIND INTERESSANT
COMPUTER     SIND         INTERESSANT
             ↑            ↑
             Pos. 2       Pos. 3
```

41. Sie haben sich wahrscheinlich schon selbst gedacht, daß auch ein READ-Statement dazu verwendet werden kann, um String-Variablen Strings zuzuweisen. Schreiben Sie bitte hin, wie Sie sich den RUN für das folgende Programm vorstellen, das die Namen der Mitglieder des Computer-Clubs ausdruckt.

```
 5 DIM M$(20)
10 READ M$
20 PRINT M$
30 GOTO 10
40 DATA JERRY, BOBBY, MARY
50 DATA MIMI, KARL
```

RUN

. . . .

. . . .

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
- - - - - - - - - - - - - - - - - - - - - - - -
RUN
JERRY
BOBBY
MARY
MIMI
KARL
ERROR- 6 AT LINE 10
```

So wie bisher könnten wir noch eine ganze Zeit weitermachen. Es gibt aber noch vieles andere, was wir kennenlernen müssen. Daher einen guten Rat: Experimentieren Sie selbst mit Ihrem Computer soviel und sooft Sie können. Versuchen Sie es mit langen und mit kurzen Strings. Was geschieht beispielsweise, wenn ein String länger als 10 Zeichen ist? Mischen Sie Strings und Zahlen oder Probieren Sie es mit Leerräumen am Anfang und am Ende.

Jetzt sollen Sie schon wieder ein Programm schreiben! Sie haben sicher bei den verschiedensten Spielen einen Würfel benutzt. Wir möchten gern, daß Sie Ihr ganzes bisheriges BASIC-Wissen dazu verwenden, ein Programm zu schreiben, das einen rollenden Würfel imitiert. Nachstehend sehen Sie einen Lauf unseres eigenen Programms. Sehen Sie es sich genau an und benutzten Sie es als Anleitung zum Schreiben eines Programms, dessen Lauf ähnlich wie das nachfolgende Beispiel aussieht. Verwenden Sie Standard-Druck-Positionen. Nehmen Sie sich genügend Zeit und probieren Sie Ihre Lösung möglichst erst auf dem Computer aus, bevor Sie sich unser Programm ansehen.

```
RUN
WIEVIELE WUERFE? 7

EINS          DREI          DREI          SECHS
VIER          FUENF         ZWEI

WIEVIELE WUERFE? 2
```

```
EINS          SECHS

WIEVIELE WUERFE?              usw.
```

....

....

....

....

....

....

....

....

....

....

....

....

....

....

– –

Sehen Sie sich die Lösung erst an, wenn Sie Ihr eigenes Programm geschrieben haben.

```
100 REM***WUERFEL-SIMULATION
110 PRINT : PRINT "WIEVIELE WUERFE"; :INPUT R
120 K=1 : PRINT
200 REM***'ROLLE DEN WUERFEL' R MAL
210 S=INT(6*RND(1))+1
220 ON S GOTO 230,240,250,260,270,280
230 PRINT "EINS", : GOTO 290
240 PRINT "ZWEI", : GOTO 290
250 PRINT "DREI", : GOTO 290
260 PRINT "VIER", : GOTO 290
270 PRINT "FUENF", : GOTO 290
280 PRINT "SECHS",
290 IF K < R THEN LET K=K+1 : GOTO 210
300 PRINT : GOTO 110
```

Warum sind in Zeile 300 zwei PRINT-Statements vorgesehen? Versuchen Sie es einmal mit einem!

EIGENTEST

Arbeiten Sie bitte den folgenden Eigentest durch. Er wird Ihnen zeigen, wieviel Sie bis jetzt gelernt haben.

1. Nehmen wir einmal an, Sie sind der Computer. Vervollständigen Sie bitte den Lauf bei den nachfolgenden Programmen:

a)
```
10 READ A
20 PRINT A
30 DATA 27

RUN
..............
```

b)
```
10 READ A
20 READ B
30 PRINT A-B
40 DATA 27,15
RUN
..............
```

c)
```
10 READ A,B
20 PRINT A-B
30 DATA 27,15

RUN
..............
```

d)
```
10 READ A,B
20 PRINT A-B
30 DATA 27
40 DATA 15
RUN
..............
```

2. Schreiben Sie den Lauf der folgenden Programme auf:

a)
```
10 READ X
20 PRINT X,  <——— Komma am Ende des PRINT-Statements
30 GOTO 10
40 DATA 3,7,0,–2,5,–1,7,8,0,–3
RUN
..........................
..........................
..........................
```

b)
```
10 READ X
20 PRINT X;" "; <—— Semikolon am Ende des PRINT-Statements
30 GOTO 10
40 DATA 3,7,0,–2,5,–1,7,8,0,–3
RUN
..............................................
```

3. Vervollständigen Sie den Lauf der folgenden Programme:

a)
```
10 READ X
20 IF X >= 0 THEN PRINT "X = " ; X,   <———— Komma
30 GOTO 10
40 DATA 3,7,0,–2,5,–1,7,8,0,–3
RUN
```

. . . .

. . . .

b)
```
10 READ X
20 IF X >= 0 THEN PRINT "X = " ; X;   <———— Semikolon
30 GOTO 10
40 DATA 3,7,0,–2,5,–1,7,8,0,–3
RUN
```

. . . .

. . . .

4. Schreiben Sie bitte auf, wie der Lauf des folgenden Programms aussieht:

```
420 PRINT "ANTWORT", "ANZ. DER","PROZENT"
430 PRINT "(J/N)","ANTWORTEN"
RUN
```

. . . .

. . . .

5. Modifizieren Sie das Befragungs-Analyse-Programm von Abschnitt 19 so, daß die Ergebnisse wie folgt ausgedruckt werden:

```
ANTWORT      ANZ. DER      PROZENT
(J/N)        ANTWORTEN
JA           23            46
NEIN         27            54
SUMME        50            100
```

Hinweis:
Schreiben Sie das Programm um, beginnend mit Zeile 420. Die gewünschte Information wird in drei Spalten ausgedruckt, wodurch die drei ersten Standard-Druck-Positionen belegt werden.

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

6. Schreiben Sie, unter Verwendung von READ- und DATA-Statements, ein Programm, um die bei einer Umdrehung von Fahrradreifen mit verschiedenen Durchmessern zurückgelegten Strecken zu berechnen. Verwenden Sie das folgende DATA-Statement mit einer Zeilennummer nach Ihrer Wahl.

DATA 16,20,24,26,27 ←——— Reifen-Durchmesser

Nachfolgend finden Sie einen Lauf unseres Programms:

```
RUN

RADDURCHMESSER: 16
RADUMFANG: 50.24

RADDURCHMESSER: 20
RADUMFANG: 62.8

RADDURCHMESSER: 24
RADUMFANG: 75.36

RADDURCHMESSER: 26
RADUMFANG: 81.64

RADDURCHMESSER: 27
RADUMFANG: 84.78

ERROR– 6 AT LINE . .   (Zeilennummer Ihres READ-Statements)
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

7. Schreiben Sie ein Programm, mit dem das Werfen einer Münze simuliert werden kann. Das Programm sollte den Computer dazu veranlassen, folgende Schritte auszuführen: W für "Wappen" und Z für "Zahl" über den ganzen Bildschirm ausdrucken, wie es der anschließende Lauf zeigt; außerdem fragen, wieviele Würfe gewünscht werden, diese Anzahl ausführen und schließlich beim Erreichen der

korrekten Anzahl stoppen. Hier sehen Sie jetzt zwei Läufe unseres Programms:

```
RUN

WIEVIELE WUERFE? 20

W W Z Z W Z Z Z Z W Z W Z Z W W W Z Z
Z


RUN

WIEVIELE WUERFE? 100

W Z Z Z W Z W Z W W W W Z Z W W Z W Z
W Z W Z Z Z Z W W W Z W Z W W Z Z W Z
W W W W Z W Z W W Z Z Z Z W Z W W Z W
W Z W Z Z W W Z Z Z Z W Z Z W Z Z W Z
W W Z W Z Z W W W Z W Z W W Z Z W Z W
W Z W W Z
```

....

....

....

....

....

....

....

....

....

....

8. Um die jeweilige Anzahl der W und Z zu ermitteln, ist es am einfachsten, man übergibt dem Computer diese Aufgabe, Modifizieren Sie dazu das Programm aus Abschnitt 7 in geeigneter Weise. Verwenden Sie die Variable W zum Zählen der Anzahl der "Wappen" und die Variable Z zum Zählen der Anzahl der "Zahlen". Ein Lauf des modifizierten Programms sollte etwa so aussehen:

```
RUN

WIEVIELE WUERFE? 20

W Z Z Z W Z W Z W W Z W W W Z W Z W Z
W

11 MAL WAPPEN UND 9 MAL ZAHL
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

9. Schreiben Sie ein Programm zum Zählen der Daten in allen DATA-Statements eines Programms. Zählen Sie aber das Flag nicht mit! Wenn sich beispielsweise die folgenden DATA-Statements im Programm befinden, sollte uns der Computer mitteilen, daß insgesamt 25 Zahlen vorhanden sind.

```
150 DATA 2,3,5,7,11,13,17,19,23,29,31,37,41
160 DATA 43,47,53,59,61,67,71,73,79,83,89,97,1E97
```

Ein Lauf unseres Programms sieht so aus:

```
RUN
ES SIND 25 DATEN VORHANDEN
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

10. Schreiben Sie bitte ein Programm, das ein Muster von Sternen (*) ausdruckt, und zwar nach einem Plan, der in Form von Werten für ein DATA-Statement eingegeben wurde. Verwenden Sie ein Flag am Ende der Daten, um eine Fehlermeldung zu vermeiden. Anschließend sehen Sie die DATA-Statement-Werte und das von unserem Programm erzeugte Muster. Unser Programm druckt fünf verschiedene Zeilen aus Sternen und Zwischenräumen.
Werte für das DATA-Statement:

```
3,4,4,5,1,5,1,5,1,5,2,2,2,5,1,5,1,5,1,5,4,4,3
RUN

*********************
* * * * * * * * * * *
* * * * * * * * * * *
***   ***   ***   ***
*******  ***  *******
***   ***   ***   ***
*******  ***  *******
***   ***   ***   ***
*******  ***  *******
***   ***   ***   ***
*** *****   ***** ***
*** *****   ***** ***
*** *****   ***** ***
***   ***   ***   ***
*******  ***  *******
***   ***   ***   ***
*******  ***  *******
***   ***   ***   ***
*******  ***  *******
***   ***   ***   ***
* * * * * * * * * * *
* * * * * * * * * * *
*********************
```

Hier ein kleiner Tip: Das (Muster besteht aus fünf verschiedenen Zeilenarten. Die erste und die letzte Zeile gehören zum "Typ 3". Die erste und die letzte Zahl im DATA-Statement lauten ebenfalls 3. Haben Sie jetzt eine Idee?

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Antworten zum Eigentest

Die Zahlen in Klammern geben die Abschnitte an, in denen der jeweilige Stoff behandelt wurde.

1. a) 27 b) 12 c) 12 d) 12
(Abschnitt 1 – 6)

2. a)
```
3     7     0    -2
5    -1     7     8
0    -3
ERROR- 6 AT LINE 10
```
b)
```
3  7  0  -2  5  -1  7  8  0  -3
```
(Abschnitt 1 – 5)

3. a)
```
X = 3     X = 7     X = 0     X = 5
X = 7     X = 8     X = 0
ERROR- 6 AT LINE 10
```
b)
```
X = 3X = 7X = 5X = 7X = 8X = 0
ERROR- 6 AT LINE 10
```
Wie würden Sie Zeile 20 abändern, um mehr Zwischenraum zwischen den einzelnen Ergebnissen zu haben und der Ausdruck bsispielsweise so aussieht?

X = 3 X = 7 usw. (Abschnitt 32, 33)

4.
```
ANTWORT     ANZ. DER      PROZENT
(J/N)       ANTWORTEN
```
(Abschnitt 35)

5.
```
420 PRINT "ANTWORT", "ANZ. DER", "PROZENT"
430 PRINT "(J/N)","ANTWORTEN"
440 PRINT
450 LET S=J+N
460 PRINT "JA", J,100*J/S
470 PRINT "NEIN",N,100*N/S
480 PRINT "SUMME",S,100
```

Im obigen Programm haben wir S benutzt, um die Summe der abgegebenen Stimmen (S = J + N) zu zählen. In den Zeilen 460 und 470 wird der Prozentsatz der Ja- und Nein-Stimmen durch die Ausdrücke 100*J/S und 100*N/S berechnet. (Abschnitte 18 – 20, 35)

6.
```
100 REM***UMFANG EINES RADES
110 READ D
120 PRINT
130 PRINT "RADDURCHMESSER:" ; D
140 PRINT "RADUMFANG:" ; 3.14*D
150 GOTO 110
160 DATA 16,20,24,26,27
```
(Abschnitt 1)

7.
```
100 REM***MUENZENWURF-SIMULATION
110 PRINT : PRINT "WIEVIELE WUERFE" ; : INPUT N
120 IF N < 1 THEN END
130 REM***WERFE DIE MUENZE N MAL
140 K=0 : PRINT
150 C=INT(2*RND(1))
160 IF C=0 THEN PRINT "W";
170 IF C=1 THEN PRINT "Z";
180 K=K+1
190 IF K < N THEN 150
200 END
```

Es gibt mehrere Möglichkeiten, dieses Programm zu schreiben. In unserer Lösung verwenden wir die Variable K, um die Anzahl der Münzenwürfe zu ermitteln. In Zeile 140 setzen wir K auf 0. Bei jedem Schleifendurchlauf wird K um 1 erhöht (Zeile 180) und mit N (Zeile 190) verglichen. Solange K kleiner als N ist, erhöhen wir jeweils K um 1 und durchlaufen nochmals die Schleife. Ist N kleiner als 1, erfolgt kein Wurf mehr. Das wird in Zeile 120 geprüft. Ihr Programm kann völlig anders aussehen, aber wenn es arbeitet, haben Sie das Problem gelöst. (Abschnitte 37, 42)

8. Wir haben das Programm aus Frage 7 in geeigneter Weise abgeändert. Das vollständige Programm ist nachstehend aufgelistet. Die Variable W wird verwendet, um die Anzahl der "Wappen" und die Variable "Z" um die Anzahl der "Zahlen" zu ermitteln.
Achten Sie auf W und Z in den Zeilen 140, 160, 170 und 210.

```
100 REM***MUENZENWURF-SIMULATION
110 PRINT : PRINT "WIEVIELE WUERFE"; : INPUT N
120 IF N < 1 THEN END
130 REM***WERFE DIE MUENZE N MAL
140 K=0 : W=0 : Z=0 : PRINT
150 C=INT(2*RND(1))
160 IF C=0 THEN PRINT "W"; : W=W+1
170 IF C=1 THEN PRINT "Z"; : Z=Z+1
180 IF K < N THEN 150
200 PRINT : PRINT
210 PRINT H:" WAPPEN UND ";T;" MAL ZAHL"
220 END
```

(Abschnitte 33, 37, 42)

9.
```
100 REM***ZAEHLE DIE DATEN IN DATA–STATEMENTS,
    AUSSER DEM FLAG
110 K=0
120 READ N
130 IF N <> 1E97 THEN K=K+1 : GOTO 120
140 IF N=1E97 THEN PRINT "ES SIND ";K;" DATEN
    VORHANDEN."
150 DATA 2,3,5,7,11,13,17,19,23,29,31,37,41
160 DATA 43,47,53,59,61,67,71,73,79,83,89,97,1E97
999 END
```

(Abschnitte 10, 11, 19)

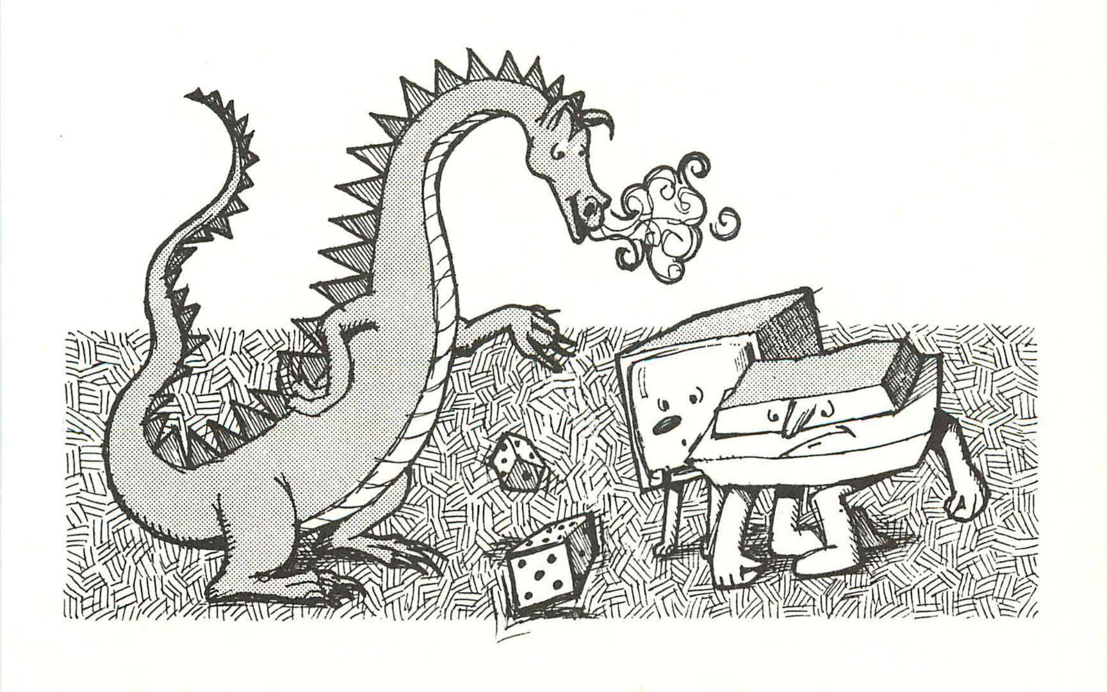
10.
```
100 REM***PROGRAMM ZUM DRUCKEN EINES MUSTERS
110 READ X : IF X=–1 THEN END
120 ON X GOTO 210,220,230,240,250
200 REM***MUSTERZEILEN
210 PRINT "*******   ***   ********" : GOTO 110
220 PRINT "*** *****      ***** ***" : GOTO 110
230 PRINT "*********************" : GOTO 110
240 PRINT "*  *  *  *  *  *  *  *  *  *  *" : GOTO 110
250 PRINT "***     ***     ***     ***" : GOTO 110
300 DATA 3,4,4,5,1,5,1,5,1,5,2,2,2,5,1,5,1,5,4,4,3,–1
```

Hier sehen Sie noch einen anderen Lösungsweg:

```
10 REM***PROGRAMM ZUM DRUCKEN VON MUSTERN
30 IF X=–1 THEN END
40 IF X=1 THEN 100
50 IF X=2 THEN 120
```

```
 60 IF X=3 THEN 140
 70 IF X=4 THEN 160
 80 IF X=5 THEN 180
 90 GOTO 20
100 PRINT "*******    ***    *******"
110 GOTO 20
120 PRINT "*** *****      ***** ***"
130 GOTO 20
140 PRINT "*********************"
150 GOTO 20
160 PRINT "* * * * * * * * * * *"
170 GOTO 20
180 PRINT "***    ***    ***    ***"
190 GOTO 20
900 DATA 3,4,4,5,1,5,1,5,1,5,2,2
910 DATA 2,5,1,5,1,5,1,5,4,4,3,-1
```

(Abschnitt 18, 19)



NOTIZEN

KAPITEL 6

FOR-NEXT-Schleifen

In diesem Kapitel möchten wir Sie mit einem anderen wichtigen Konzept der Computer-Programmierung vertraut machen: der FOR-NEXT-Schleife. Das IF-THEN-Statement und die FOR-NEXT-Schleife erweitern die Anwendungsmöglichkeiten unseres Computers beträchtlich. Wenn Sie aufmerksam den Erklärungen in diesem Kapitel folgen und die jeweiligen Fragen beantworten, werden Sie die Funktion dieser BASIC-Statements ohne Schwierigkeiten verstehen und mit Ihren Programmierkenntnissen ein beträchtliches Stück vorankommen. Wenn Sie dieses Kapitel beendet haben, können Sie . . .

- die Statements FOR und NEXT benutzen, und
- die STEP-Bestimmung in FOR-Statements anwenden.

1. Das folgende Schleifen-Demonstrations-Programm ist ein Zählprogramm, das die Werte von F ausdruckt, wenn wir F von 1 bis 6 erhöhen. Zeiie 10 dient lediglich dazu, um F mit 1 zu initialisieren. Initialisierung bedeutet, wie Sie sich sicherlich erinnern, einer Variablen den ersten Wert zuzuweisen. Zeile 20 druckt den gegenwärtigen Wert von F aus. Zeile 30 erhöht F bei jedem Programmdurchlauf um 1.

Zeile 40 enthält ein IF-Statement, das den Wert von F abfragt. Solange F kleiner oder gleich 6 ist, schickt Zeile 40 den Computer wieder zurück zur Zeile 20, um den Wert von F auszudrucken. Sobald F größer als 6 wird, stoppt das Programm, da keine weiteren Statements mehr vorhanden sind. Schreiben Sie das Programm zur Übung um, und zwar unter Verwendung von Mehrfach-Statements in einer Zeile. Überlegen Sie bitte sorgfältig, ob Sie das ganze Programm in einer Zeile schreiben können.

```
 5 REM***SCHLEIFEN-DEMONSTRATION
10 LET F=F+1
20 PRINT "F = "; F
30 LET F=F+1
40 IF F <= 6 THEN 20
```

```
RUN
F = 1
F = 2
F = 3
F = 4
F = 5
F = 6
```

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

```
10 LET F = 1
20 PRINT "F = "; F : LET F=F+1 : IF F <= 6 THEN 20
```

Das IF-THEN-Statement in Zeile 20 bedeutet: Ist der Vergleich in Zeile 20 WAHR, gehe wieder an den Zeilenanfang zurück und führe nochmals alle Statements aus. Zeile 10 würde zwar ebenfalls noch in diese Zeile hineinpassen, jedoch muß dabei beachtet werden, daß in diesem Fall der Wert für F jedesmal bei der Ausführung des Statements initialisiert würde. Daher können wir nicht das ganze Statement in eine Zeile schreiben.

2. Die Platz und Zeit sparende FOR-NEXT-Schleife führt in sehr einfacher Weise eine vorgegebene Anzahl von Wiederholungen oder Schritten aus. Sehen Sie sich einmal die nachfolgende FOR-NEXT-Schleife an. Statt IF-THEN verwenden wir diesmal FOR- und NEXT-Statements, um dem Computer zu sagen, wie oft er die Schleife durchlaufen muß.

```
 5 REM***FOR-NEXT-SCHLEIFEN-DEMONSTRATION
10 FOR F=1 TO 6
20 PRINT "F = "; F
30 NEXT F
```

In jeder FOR-NEXT-Schleife ist das FOR-Statement der Anfang und das NEXT-Statement immer das letzte Statement der Schleife. Das oder die Statements zwischen FOR und NEXT werden in der vorgegebenen Reihenfolge immer wieder ausgeführt, wobei das FOR-Statement dem Computer anzeigt, wie oft die Schleife durchlaufen werden muß.

a) Sie sehen aus dem Ablauf der FOR-NEXT-Schleife, daß der Wert von F bei jedem Schleifendurchlauf automatisch um . . erhöht wird.

b) Wie oft hat der Computer die Schleife durchlaufen? . . . .

c) Warum hielt der Computer an, nachdem er die angegebene Zahl von Schleifen ausgeführt hatte? . . . .

d) Welches andere Statement brauchen Sie noch für die Schleife? . . . . .

— — — — — — — — — — — — — — — — — — —

a) 1 b) 6
c) Weil ihn das FOR-Statement anwies, von 1 bis 6 zu gehen.
d) Ein NEXT-Statement

3. Wie Sie an dem folgenden Programm sehen können, fährt der Computer mit dem Rest des Programms fort, sobald er die durch das FOR-Statement angegebenen Schleifen ausgeführt hat.

```
 5 REM***NOCH EINE FOR-NEXT-SCHLEIFE
10 FOR D=5 TO 10◄———Die Schleife muß nicht mit 1 beginnen!
20 PRINT "D = "; D
30 NEXT D
40 PRINT
50 PRINT "AHA! AUSSERHALB DER SCHLEIFE, DA"
60 PRINT "D = " ; D: "UND DAMIT GROESSER ALS 10 IST."

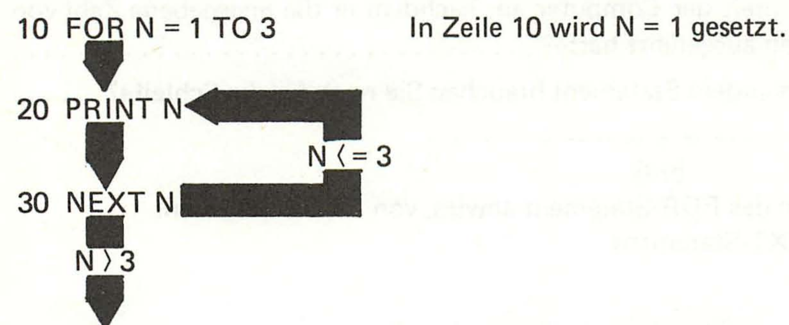
RUN
D = 5
D = 6
D = 7
D = 8
D = 9
D = 10
AHA! AUSSERHALB DER SCHLEIFE, DA
D = 11 UND DAMIT GROESSER ALS 10 IST.
```

Welche Zeilen belegt die FOR-NEXT-Schleife in dem Programm? . . . . .
. . . .

— — — — — — — — — — — — — — — — — —

Zeilen 10, 20 und 30

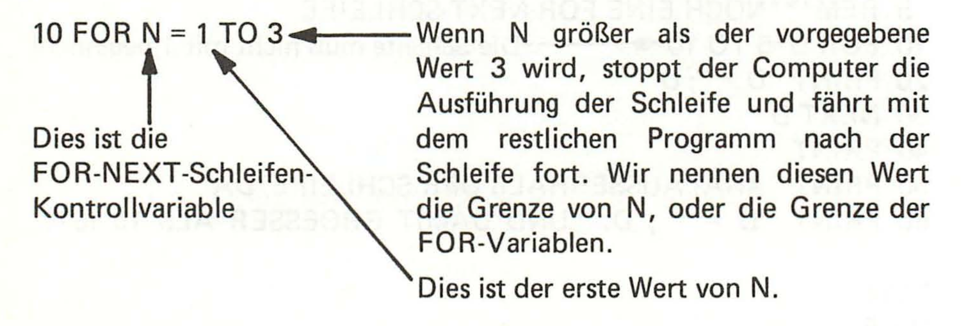
4. Wie arbeitet die FOR-NEXT-Schleife? Folgen Sie den Pfeilen:



In Zeile 10 wird N = 1 gesetzt.

In Zeile 30 wird N um 1 erhöht. Dann vergleicht der Computer den neuen Wert von N mit der im FOR-NEXT-Statement angegebenen oberen Grenze für N.

Sehen wir uns das FOR-NEXT-Statement einmal näher an:



Und so sieht ein Lauf des obigen Programms aus:

```
10 FOR N=1 TO 3
20 PRINT N
30 NEXT N
RUN
1
2
3
```

Jedesmal, wenn der Computer auf ein NEXT-N-Statement trifft, erhöht er den Wert von N um 1 und vergleicht den neuen Wert mit der Grenze von N. In diesem Fall ist die Grenze 3, da das FOR-Statement lautet: FOR N = 1 TO 3. Sobald der Wert von N größer als 3 wird,

fährt der Computer mit dem nächsten Statement nach dem NEXT-Statement fort, sofern noch eins vorhanden ist. Ist das nicht der Fall, hat er die Programmausführung beendet und hält an. Haben Sie alles verstanden? Sie können es gleich selbst nachprüfen!

Statement 10 bedeutet, daß beim ersten Schleifendurchlauf N=1 ist. Beim zweiten Schleifendurchlauf ist N= N + 1 = 1 + 1 = 2. Beim dritten Durchlauf schließlich ist

N = ... = ... = ...

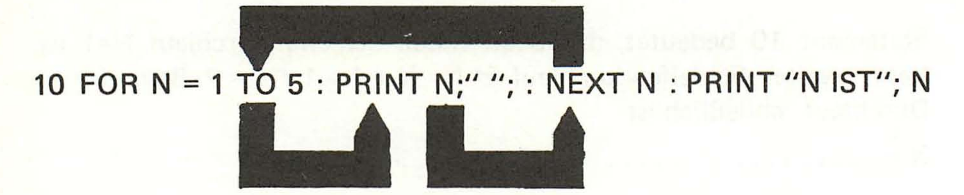
N = N + 1 = 2 + 1 = 3

5. Schreiben Sie ein Programm mit drei Statements, welches das Wort SCHLEIFE sechsmal druckt. Verwenden Sie dazu eine FOR-NEXT-Schleife und benutzen Sie C als die FOR-NEXT-Schleifen-Kontroll-Variable. Schreiben Sie bitte auf, was Ihr Programm ausdruckt, wenn es abläuft.

....
....
....
....
....
....
....
....
....
....

```
10 FOR C = 1 TO 6
20 PRINT "SCHLEIFE"
30 NEXT C
RUN
SCHLEIFE
SCHLEIFE
SCHLEIFE
SCHLEIFE
SCHLEIFE
SCHLEIFE
```

6. FOR-NEXT-Statements können auch in Zeilen mit mehreren Statements vorgesehen werden. Hier sehen Sie ein ganzes FOR-NEXT-Programm in einer einzigen Zeile. Folgen Sie zum Verständnis den angegebenen Pfeilen.



```
10 FOR N = 1 TO 5 : PRINT N;" "; : NEXT N : PRINT "N IST"; N
```

```
RUN
1       2       3       4       5
N IST 6
```

Wie groß war der Wert von N, als der Computer die FOR-NEXT-Schleife beendet und sie verlassen hatten? ....

\- -

6

Beachten Sie bitte, daß der Wert von N nach dem Verlassen der Schleife um 1 größer ist als die im FOR-Statement angegebene obere Grnze für N. Auf diese Weise stellt der Computer fest, daß er die geforderte Anzahl von Schleifen ausgeführt hat und zum nächsten Statement im Programm weitergehen kann.

7. Nehmen wir einmal an, Sie geben die folgende Statement-Zeile ein.

```
10 PRINT "TEST"; : FOR X = 1 TO 4 : PRINT X; " "; : NEXT X
```

Was wird Ihr Computer ausdrucken, wenn Sie die Ausführung durch RUN starten? ....

\- -

```
TEST  1  2  3  4
```

8. Vervollständigen Sie das nachfolgende Programm bitte so, daß der Computer den angegebenen Lauf ausführt. Verwenden Sie eine FOR-NEXT-Schleife mit Y als Kontroll-Variable.

```
10 PRINT "IN DIESEM JAHR IST CATHY 10 JAHRE ALT"
20 ..................................................
30 ..................................................
40 ..................................................
```

```
RUN
IN DIESEM JAHR IST CATHY 10 JAHRE ALT
UND IM NAECHSTEN JAHR WIRD SIE 11
UND IM NAECHSTEN JAHR WIRD SIE 12
UND IM NAECHSTEN JAHR WIRD SIE 13
UND IM NAECHSTEN JAHR WIRD SIE 14
```

– – – – – – – – – – – – – – – – – –

```
20 FOR Y = 11 TO 14
30 PRINT "UND IM NAECHSTN JAHR WIRD SIE"; Y
40 NEXT Y
```

9. Es wäre noch darauf hinzuweisen, daß man bei FOR-NEXT-Schleifen auch die Variablen statt der numerischen Werte einsetzen kann, vorausgesetzt natürlich, daß den Variablen zuvor im Programm Werte zugewiesen wurden. Im folgenden Beispiel werden die Werte, wie üblich, durch LET-Statements zugewiesen. Die Werte hätten auch durch INPUT- oder READ-Statements zugewiesen werden können.

```
10 A=3 : B=8
20 FOR C=A TO B
30 PRINT " "; C;
40 NEXT C
RUN
3 4 5 6 7 8
```

Schreiben Sie das FOR-Statement in Zeile 20 um, und ersetzten Sie dabei die numerischen Werte für die Variablen A und B. Verwenden Sie die Werte, die im obigen Programm zugeordnet wurden.

```
20 ..................................................
```

– – – – – – – – – – – – – – – – – – –

```
20 FOR C = 3 TO 8
```

10. Spielen Sie einmal Computer und geben Sie den Lauf für dieses FOR-Next-Demonstrations-Programm an.

```
10 X = 0 : Y=4
20 FOR Z=X TO Y
40 PRINT " "; Z
50 NEXT Z
```

RUN

. . . .

\- -

0 1 2 3 4

11. Schreiben Sie jetzt ein Programm, in dem die Variablen dazu verwendet werden, den Anfangs- und Endwert der FOR-NEXT-Schleifen-Kontrollvariablen festzulegen. Schreiben Sie Ihr Programm so, daß die benötigten Daten durch READ aus einem DATA-Statement gelesen werden.

Verwenden Sie Z als Kontrollvariable und X und Y für die Variablen, welche die Anfangs- und Endwerte festlegen. Wählen Sie die Werte für Ihr DATA-Statement so, daß das Programm den folgenden Lauf erzeugt:

RUN
100 101 102 103 104 105

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

\- -

```
10 READ X
20 READ Y
30 FOR Z=X TO Y
40 PRINT " "; Z
50 NEXT Z
60 DATA 100,105
```

Die Statements 10 und 20 könnnten auch zu einem einzigen zusammengefaßt werden:

10 READ X,Y

12. Im folgenden Programm wird ein durch INPUT (Zeile 40) eingegebener Wert verwendet, um die obere Grenze des FOR-NEXT-Statements (Zeile 100) festzulegen, die dem Computer angibt, wie oft er "X = ?" wiederholen soll. Wenn das Programm läuft, sagen die PRINT-Statements in den Zeilen 10 bis 80 dem Benutzer, wie er mit dem Programm arbeiten muß.

```
  5 REM***MITTELWERTPROGRAMM
 10 PRINT "ICH WERDE DEN MITTELWERT EINER LISTE"
 20 PRINT "VON ZAHLEN BERECHNEN."
 30 PRINT
 40 PRINT"WIEVIELE ZAHLEN ENTHAELT DIE LISTE?":INPUTN
 50 PRINT
 60 PRINT "JEDESMAL WENN ICH 'X =?' FRAGE' GEBEN SIE"
 70 PRINT "EINE ZAHL EIN UND DRUCKEN AUF RETURN."
 80 PRINT
 90 T=0
100 FOR K=1 TO N
110 PRINT "X = "; : INPUT X
120 T=T+X
130 NEXT X
140 M=T/N
150 PRINT
160 PRINT "SUMME = "; T
170 PRINT "MITTELWERT = "; M
```

a) Im obigen Programm belegt die FOR-NEXT-Schleife die Zeilen ....

b) Wie lautet die FOR-NEXT-Schleifen-Kontrollvariable? ....

c) Die obere Grenze der Kontrollvariablen wird durch einen Wert festgelegt, der welcher anderen Variablen zugewiesen wurde? ....

– –

a) 100, 110, 120, 130 b) K c) N

13. Welche Zeile der FOR-NEXT-Schleife im Programm aus Abschnitt 12 enthält einen laufenden Zähler für die für Zeile 110 eingegebenen Werte? ....

----

120 T = T + X

14. Dies ist ein Lauf des Programms aus Abschnitt 12:

RUN

ICH WERDE DEN MITTELWERT EINER LISTE
VON ZAHLEN BERECHNEN.

WIEVIELE ZAHLEN ENTHAELT DIE LISTE? 5

JEDESMAL WENN ICH 'X=?' FRAGE, GEBEN SIE
EINE ZAHL EIN UND DRUCKEN AUF RETURN.

X =? 16
X =? 46
X =? 38
X =? 112
X =? 23

SUMME = 235
MITTELWERT = 47

Geben Sie bitte die numerischen Werte für das FOR-NEXT-Statement des obigen Laufs an.

100 FOR K = ... TO ....

----

1 TO 5 (Der für INPUT N eingegebene Wert war 5.)

15. Hier sehen Sie den Anfang eines weiteren Laufs des gleichen Programms.

RUN

ICH WERDE DEN MITTELWERT EINER LISTE
VON ZAHLEN BERECHNEN.

WIEVIELE ZAHLEN ENTHAELT DIE LISTE? 4 ←—Dies ist der eingegebene Wert.

Wie oft wird "X = ?" gedruckt? ....

Wie oft werden die Statements zwischen den FOR-NEXT-Statements ausgeführt? ....

4; 4

Nur um Ihnen das zu beweisen – hier ist der Rest des gleichen Laufs:

```
JEDESMAL WENN ICH 'X=?' FRAGE, GEBEN SIE
EINE ZAHL EIN UND DRUCKEN AUF RETURN.

X =? 2
X =? 4
X =? 6
X =? 8

SUMME = 20
MITTELWERT = 5
```

16. Vervollständigen Sie das nächste Programm, mit dem das Produkt (P) von N Zahlen berechnet wird. Denken Sie sorgfältig über die Wirkung Ihrer Statements beim Lauf des Programms nach.

```
  5 REM***DAS PRODUKT VON N ZAHLEN
 10 PRINT "ICH WERDE IHNEN DAS PRODUKT EINER"
 20 PRINT "LISTE VON ZAHLEN BERECHNEN."
 30 PRINT
 40 PRINT "WIEVIELE ZAHLEN HAT DIE LISTE"; : INPUT N
 50 PRINT
 60 PRINT "JEDESMAL WENN ICH 'X =?' FRAGE, GEBEN SIE EINE"
 70 PRINT "ZAHL EIN UND DRUECKEN DANN AUF RETURN."
 80 PRINT
100 ..................... <-------------------- Initialisierung
110 ...................................................
120 PRINT "X = "; : INPUT X
130 P=P*X
140 ...................................................
150 PRINT :PRINT "PRODUKT ="; P

RUN
ICH WERDE IHNEN DAS PRODUKT EINER
LISTE VON ZAHLEN BERECHNEN.

WIEVIELE ZAHLEN HAT DIE LISTE? 5
```

```
JEDESMAL WENN ICH 'X =?' FRAGE, GEBEN SIE EINE
ZAHL EIN UND DRUECKEN DANN AUF RETURN.
X =? 7
X =? 12
X =? 4
X =? 4
X =? 3
X =? 19
PRODUKT = 19152
```

```
100 LET P=1
110 FOR K=1 TO N
140 NEXT K
```

17. Zur Festlegung sowohl des Anfangs- als auch des Endwertes einer FOR-NEXT-Schleifen-Kontrollvariablen kann jeder beliebige BASIC-Ausdruck verwendet werden. Der Computer berechnet diese Ausdrücke, bevor die Schleife zum ersten Mal ausgeführt wird. Während der einzelnen Schleifendurchläufe berechnet er sie nicht mehr neu.

Sehen Sie sich das FOR-Statement im nächsten Programm einmal genau an und entscheiden Sie dann, ob der Computer die korrekte Anzahl von Schleifendurchläufen ausführte.

```
10 Q=4
20 FOR P=Q TO 2*Q–1
30 PRINT " "; P;
40 NEXT P
RUN
4  5  6  7
```

Der Computer hatte recht!

Füllen Sie bitte im folgenden Programm die Leerstellen in Zeile 20 mit Ausdrücken unter Verwendung der Variablen Q aus, so daß das Programm beim Ablauf den folgenden Ausdruck erzeugt:

```
10 Q=4
20 FOR P= ...... TO ......
30 PRINT " "; P;
```

```
40 NEXT P
RUN
2 3 4 5 6 7 8 9 10 11 12
```

```
20 FOR P=Q/2 TO Q*3 oder 20 FOR P=Q–2 TO Q+8
```

Hinweis:
Sollte Ihre Lösung anders aussehen und Sie sind der Ansicht, daß sie richtig ist, dann probieren Sie sie auf Ihrem Computer aus, um zu sehen, ob Sie den gleichen Lauf wie wir erhalten.

18. In den FOR-NEXT-Schleifen, die Sie bisher gesehen haben, übernimmt die FOR-NEXT-Schleifen-Kontrollvariable den ersten, im FOR-Statement angegebenen Wert und behält Ihn solange bei, bis der Computer zum NEXT-Statement kommt. Dann erhöht die FOR-Variable bei jedem Schleifen-Durchlauf ihren Wert, bis sie den maximal möglichen erreicht hat, den das FOR-Statement zuläßt.

FOR X 5 TO 10

Erster Wert von X — Maximaler Wert für X.

X ist zuerst also 5 und nimmt dann die Werte 6, 7, 8, 9, und schließlich 10 an.

Es ist jedoch möglich, ein FOR-Statement zu schreiben, das ein Ansteigen des Wertes der FOR-NEXT-Schleifen-Kontrollvariablen um Vielfache beliebiger anderer Zahlen als 1 oder in Bruchteilen von 1 veranlaßt.

Es ist auch möglich, die Schleifen-Kontrollvariable bei jedem Schleifendurchlauf zu verringern:

```
10 FOR X=1 TO 10 STEP 2
```

STEP 2 weist den Computer an, den Wert von X bei jedem Schleifendurchlauf um 2 zu erhöhen, bis X größer als 10 ist.

```
10 FOR Y=3 TO 6 STEP 1.5
```

STEP 1.5 weist den Computer an, den Wert von Y bei jedem Schleifendurchlauf um 1.5 zu erhöhen, bis Y größer als 6 ist.

```
10 FOR Z=10 TO 5 STEP -1
```

Beachten Sie, daß Z bei 10 beginnt und bis auf 5 verringert wird.

STEP −1 weist den Computer an, den Wert von Z bei jedem Schleifendurchgang um 1 zu verrringern, bis er kleiner als 5 ist.

Einige Demonstrationsprogramme sollen Ihenen die praktische Wirkung von STEP zeigen.

```
10 FOR B=1 TO 10 STEP 2
20 PRINT " "; B;
30 NEXT B
40 PRINT
50 PRINT
60 PRINT "DIE SCHLEIFE IST BEENDET, DA"
70 PRINT "B = "; B; "UND DAMIT GROESSER ALS 10 IST."
RUN
1 3 5 7 9
DIE SCHLEIFE IST BEENDET, DA
B = 11 UND DAMIT GROESSER ALS 10 IST.
```

Beachten Sie, daß die Schleife mit dem ersten Wert des FOR-Statements beginnt (1) und in Schritten von 2 anwächst, bis der Wert von B=11 ist, und damit die vorgegebene Grenze von 10 überschreitet. An diesem Punkt beendet der Computer die Schleife und fährt mit dem Rest des Programms fort.

Spielen Sie jetzt wieder einmal Computer und füllen Sie bitte den Lauf für das folgende Programm aus.

```
10 D=3 : FOR F=D TO 4*D STEP D : PRINT " "; F; : NEXT F
RUN
..................................................
```

---

3 6 9 12

19. In diesem Beispiel folgt auf STEP im FOR-Statement eine negative Zahl. STEP kann dazu verwendet werden, den Wert der FOR-Variablen in Schritten mit beliebiger Größe zu verringern.

```
10 FOR J=100 TO 10 STEP -10
20 PRINT " "; J;
30 NEXT J
RUN
100 90 80 70 60 50 40 30 20 10
```

Schreiben Sie jetzt ein Programm, in dem die FOR-NEXT-Kontrollvariable E in Schritten von 3 von 27 auf 18 verringert wird. Schreiben Sie das Programm und einen Lauf.

....

....

....

....

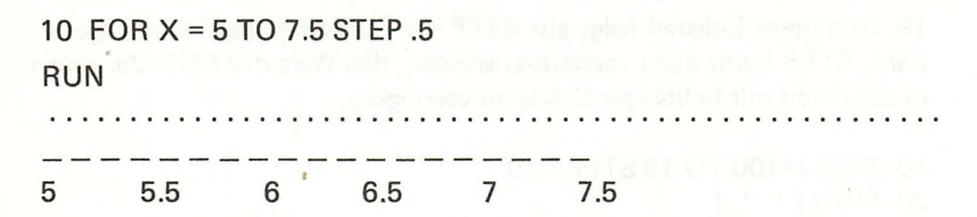
....

```
10 FOR E = 27 TO 18 STEP -3
20 PRINT " "; E;
30 NEXT E
RUN
27 24 21 18
```

20. Wir haben bereits darauf hingewiesen, daß die Schritte in einer FOR-NEXT-Schleife auch Brüche sein können. Das folgende Beispiel soll dies demonstrieren.

```
10 FOR X = 5 TO 7.5 STEP .25
20 PRINT " "; X;
30 NEXT X
RUN
5   5.25   5.5   5.75   6   6.25   6.5   6.75   7   7.25   7.5
```

Schreiben Sie bitte einen Lauf für dieses Programm, wenn wir Zeile 10 folgendermaßen ändern.



```
10 FOR X = 5 TO 7.5 STEP .5
RUN
.........................................................
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
5       5.5       6       6.5       7       7.5
```

21. Die FOR-NEXT-Schleife ist zweckmäßig für Probleme wie wiederholte Berechnungen, Zählschleifen oder die Behandlung zyklischer Ereignisse.

Ein derartiges, immer wiederkehrendes Ergebnis ist die monatliche Berechnung der Zinsen für ein Sparkonto. Im folgenden Programm werden die monatlichen Zinsen (Z) in Zeile 180 durch Multiplikation des Sparkapitals (K) mit dem Zinssatz (P) ermittelt.

Der Zinssatz wird in einen Dezimalbruch umgewandelt: P = 5 Prozent = 5/100 = .05.

Da 5 Prozent der jährliche Zinssatz sind, wird nur 1/12 der berechneten Zinsen monatlich zum Sparkapital hinzuaddiert.

```
100 REM***MONATLICHE ZINSBERECHNUNG
110 PRINT "SPARKAPITAL" ; : INPUT K
120 PRINT " JAEHRLICHER ZINSSATZ (IN %)" ; : INPUT P
130 PRINT "WIEVIELE MONATE"; : INPUT M
140 PRINT
150 PRINT "MONAT", "BETRAG"
160 FOR K=1 TO M
170 PRINT K, P
180 Z=(K*(P/100)) /12
190 K=K+Z
220 NEXT K
RUN
SPARKAPITAL? 200
JAEHRLICHER ZINSSATZ (IN %)? 5
```

```
WIEVIELE MONATE? 6
MONAT        BETRAG
1            200
2            200.833333
3            201.670138
4            202.51043
5            203.354223
6            204.201532
```

Diese Tabelle gibt den Stand des Sparkontos jeweils am Beginn eines Monats an.

a) Welche Zeilen liegen innerhalb der FOR-NEXT-Schleife? . . . .

b) Welche Variable wird zum Ausdrucken der Zahl des jeweiligen Monats am Anfang einer Zeile der Tabelle verwendet? . . . .

c) Zeile 150 druckt die Spalten-Überschriften für die Tabelle. Die in den Überschriften verwendeten Worte sind durch Komma getrennt. In Zeile 170 sind die Werte, die unter den Überschriften augedruckt werden sollen, ebenfalls durch Kommas getrennt, so daß die Abstände der Überschriften und der darunter stehenden Zahlen zusammenpassen. Was würde geschehen, wenn das Statement, das die Überschrift druckt, in der FOR-NEXT-Schleife enthalten wäre?
. . . .

d) Welche Zeile enthält einen laufenden Zähler für den jeweiligen Stand des Kapitals + Zinsen? . . . .

---

a) 160, 170, 180, 190, 200
b) Die FOR-Variable K
c) Die Überschrift würde bei jedem Schleifendurchlauf zwischen den einzelnen Zeilen der Tabelle gedruckt werden.
d) 190

22. Die sechs Stellen hinter dem Komma ausgedruckten Sparkapital haben Ihnen sicher schon gestört. Glücklicherweise gibt es in BASIC eine einfache Möglichkeit, um Zahlen bis auf eine gewünschte Stellenzahl hinter dem Komma abzurunden.

Erinnern Sie sich noch daran, wie die INT-Funktion arbeitet? (Kapitel

4, Abschnitt 20). Die INT-Funktion schneidet eine Zahl am Dezimalpunkt ab, behält den ganzzahligen Anteil übrig und vernachlässigt den Dezimalbruchanteil. Zum Beispiel: INT (200.833333) = 200

Da wir es aber mit DM und Pfennig zu tun haben, möchten wir natürlich keineswegs auf die Pfennigangabe verzichten. Daher gehen wir folgendermaßen vor:

1) Multiplikation mit 100	10*200.833333 = 20083.333
2) Addition von .5	20083.333 + .5 = 20083.8333
3) ganzzahliger Anteil	INT (20083.8333) = 20083
4) Division durch 100	20083/100 = 200.83

Jetzt sind Sie an der Reihe. Verwenden Sie das obige Verfahren um 123.45678 DM auf den nächsten Pfennig zu runden.

1) Mulitplikation mit 100	100*123.45678 = ...
2) Addition von 0.5	....
3) ganzzahliger Anteil	....
4) Division durch 100	....

---

1) 100*123.456789 = 123 = 12345.678
2) 12345.678 + .5 = 12346.178
3) INT(12346.178) = 12346
4) 12346/100 = 123.46

23. Das folgende Programm ist eine Schritt-für-Schritt-Demonstration der im vorigen Abschnitt gezeigten Rundungsmethode.

```
100 REM***SCHRITT-FUER-SCHRITT-DEMONSTRATION DER
    RUNDUNG
110 PRINT "ZU RUNDENDE ZAHL"; : INPUT A
120 PRINT "BEGINNEN SIE MIT IHRER ZAHL,     A = ";A
130 A = 100*A
140 PRINT "(1) MULTIPLIKATION MIT 100,      A = ";A
150 A = A + .5
160 PRINT "(2) ADDITION VON .5,             A = ",A
```

```
170 A = INT(A)
180 PRINT "(3) GANZZAHLIGER ANTEIL,          A = ";A
190 A = A/100
200 PRINT "(4) DIVISION DURCH 100,           A = ";A
210 PRINT "A IST GERUNDET AUF ZWEI DEZIMALSTELLEN"
220 PRINT
230 GOTO 110

RUN

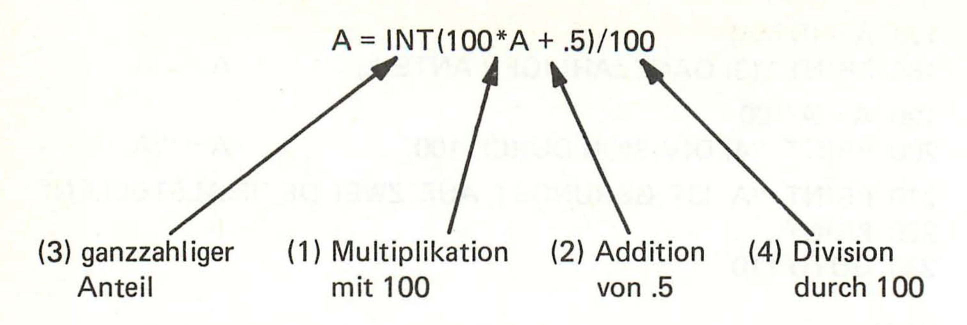
ZU RUNDENDE ZAHL, 123.45678
BEGINNEN SIE MIT IHRER ZAHL              A = 123.45678
(1) MULTIPLIKATION MIT 100,              A = 12345.678
(2) ADDITION VON .5,                     A = 12346.178
(3) GANZZAHLIGER ANTEIL,                 A = 12346.
(4) DIVISION DURCH 100,,                 A = 123.46
A IST GERUNDET AUF ZWEI DEZIMALSTELLEN
```

Hier ein weiteres Beispiel. Tragen Sie bitte die Werte von A ein und probieren Sie unser Programm auf Ihrem ATARI-Computer aus.

```
ZU RUNDENDE ZAHL               A = ......................
(1) MULTIPLIKATION MIT 100,    A = ......................
(2) ADDITION VON .5,           A = ......................
(3) GANZZAHLIGER ANTEIL,       A = ......................
(4) DIVISION DURCH 100,        A = ......................
A IST GERUNDET AUF ZWEI DEZIMALSTELLEN
```

```
    203.7249
(1) 20372.49
(2) 20372.99
(3) 23072
(4) 203.72
```

24. In unserem Rundungsprogramm erfolgt die tatsächliche Rundung in den Zeilen 130, 150, 170 und 190. Wir können diese vier Zeilen folgendermaßen zu einem einzigen Statement zum Runden einer Zahl kombinieren:



Das obige Statement rundet den Wert von A auf zwei Dezimalstellen.

1) Schreiben Sie ein Statement, um den Wert von P auf zwei Dezimalstellen abzurunden. ....

2) Schreiben Sie ein Statement, um den Wert von X auf eine Dezimalstelle zu runden. ....

3) Schreiben Sie ein Statement, um den Wert von B auf drei Dezimalstellen zu runden. ....

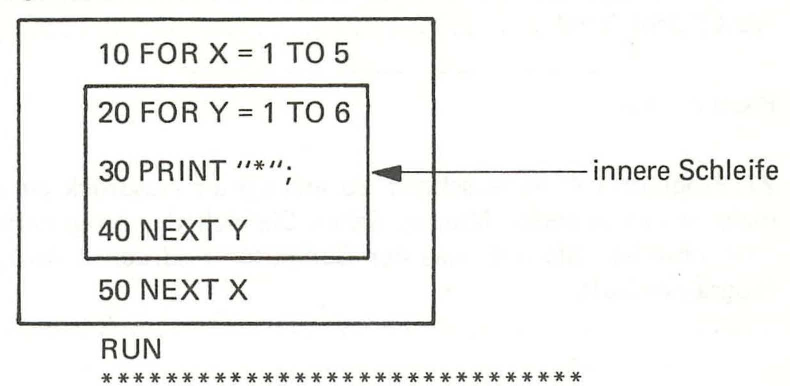
- -

1) P = INT(100*P + .5)/100
2) X = INT(10*X + .5)/10
3) B = INT(1000*B + .5)/1000

Fügen Sie jetzt bitte das Statement 195K = INT(100*K + .5)/100 in das Programm MONATLICHE ZINSBERECHNUNG aus Abschnitt 21 ein. Das sollte dazu führen, daß die Zahlen unter BETRAG auf den nächsten Pfennig gerundet werden. Ein Lauf wird dann folgendermaßen aussehen.

```
RUN
SPARKAPITAL? 200
JAEHRLICHER ZINSSATZ (IN %)? 5
WIEVIELE MONATE? 6
MONAT          BETRAG
1              200
2              200.83
3              201.67
4              202.51
5              203.35
6              204.2
```

25. Ist eine FOR-NEXT-Schleife auch innerhalb einer weiteren FOR-NEXT-Schleife möglich? Selbstverständlich! So etwas nennt man "verschachtelte Schleifen" und ist völlig korrekt, sofern Sie die nachfolgende Regel beachten:



Zu beachten ist, daß sich die innere Schleife völlig innerhalb der zweiten, äußeren Schleife befinden muß, andernfalls erhalten Sie eine Fehlermeldung.

Zählen Sie einmal die durch den Programmlauf ausgedruckten Sterne. Wieviele sind es? Sehen Sie irgend einen Zusammenhang zwischen der Anzahl der Sterne und den Zahlen 5 und 6, die in den beiden FOR-Statements enthalten sind?

Eine Schleife innerhalb einer anderen wird als .... FOR-NEXT-Schleife bezeichnet. Die innere Schleife muß vollständig .... der äußeren Schleife liegen.

\- -

verschachtelte; innerhalb

26. Hier sehen Sie zwei Programme mit verschachtelten FOR-NEXT-Schleifen.

```
 5 REM***PROGRAMM A
10 FOR N=1 TO 4
20 FOR L=1 TO 3
30 PRINT "VERSCHACHTELT"
40 NEXT L
```

```
 5 REM***PROGRAMM B
10 FOR N=1 TO 4
20 FOR L=1 TO 3
30 PRINT "VERSCHACHTELT"
40 NEXT N
```

```
50 PRINT " SCHLEIFE"          50 PRINT " SCHLEIFE"
60 NEXT N                     60 NEXT L
```

Welches Programm (A oder B) enthält korrekt verschachtelte FOR-NEXT-Schleifen? ....

Programm A

27. Programm A in Abschnitt 26 erzeugt als Ausdruck ein sich mehrmals wiederholendes Muster. Sehen Sie sich das Programm genau an und schreiben Sie auf, was der Computer ausdrucken wird, wenn das Programm läuft.

....

....

....

....

....

....

....

....

....

....

....

```
RUN
VERSCHACHTELT
VERSCHACHTELT
VERSCHACHTELT
 SCHLEIFE
VERSCHACHTELT
VERSCHACHTELT
VERSCHACHTELT
 SCHLEIFE
VERSCHACHTELT
VERSCHACHTELT
VERSCHACHTELT
 SCHLEIFE
VERSCHACHTELT
VERSCHACHTELT
```

VERSCHACHTELT
SCHLEIFE

28. Das folgende Programm verwendet verschachtelte Schleifen, um ein rechteckiges Muster von "Sternen" zu drucken.

```
10 FOR R = 1 TO 3
20 FOR C = 1 TO 7
30 PRINT "*";
40 NEXT C
50 PRINT
60 NEXT R

RUN
*******
*******
*******
```

Die innere Schleife veranlaßt den Computer, eine Reihe von 7 Sternen zu drucken.

1. Welchen Zweck hat das leere PRINT-Statement in Zeile 50? ....
   ....

2. Warum druckt der Computer drei Reihen von Sternen? ....
   ....

---

1. Es verschiebt den Cursor zum Anfang der nächsten Zeile, nachdem durch die innere Schleife eine Reihe von Sternen gedruckt wurde.
2. Die äußere Schleife, die von den Zeilen 10 und 60 kontrolliert wird, veranlaßt die Ausführung der inneren Schleife für R = 1,2 und 3. Dabei wird jedesmal eine Reihe von Sternen gedruckt.

29. Schreiben Sie auf, was gedruckt wird, wenn wir das folgende Programm laufen lassen.

```
10 FOR A=1 TO 4
20 FOR B=1 TO 3
30 PRINT "?";
40 NEXT B
50 PRINT
60 NEXT A
```

RUN

. . . .

. . . .

. . . .

. . . .

_ _

```
???
???
???
???
```

Der Computer druckt vier Reihen mit je drei Fragezeichen.

30. Schreiben Sie ein Programm, durch das 7 Reihen mit je 12 Dollar-Zeichen pro Reihe ausgedruckt werden. Ein Lauf sollte wie folgt aussehen:

```
RUN
$$$$$$$$$$$$
$$$$$$$$$$$$
$$$$$$$$$$$$
$$$$$$$$$$$$
$$$$$$$$$$$$
$$$$$$$$$$$$
$$$$$$$$$$$$
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

_ _

```
10 FOR A=1 TO 7
20 FOR Z=1 TO 12
30 PRINT "$";
40 NEXT Z
50 PRINT
60 NEXT A
```

Die von Ihnen verwendeten Variablen können natürlich anders lauten.

EIGENTEST

Nach Abschluß von Kapitel 6 müßten Sie bereits genügend Programmierkenntnisse gesammelt haben, um selbst an Ihrem Computer experimentieren zu können. Wenn Sie sich unsere Demonstrationsprogramme ansehen, dann werden Sie immer eine Reihe von Möglichkeiten entdecken, auf die wir nicht näher eingegangen sind. Darum ein guter Rat von unserer Seite: Probieren Sie hier Ihre eigenen Ideen aus, Sie werden Ihre Kenntnisse und Fähigkeiten dadurch beträchtlich erweitern.

Aber jetzt prüfen Sie erst einmal, ob Sie es wirklich verstanden haben, wie man FOR-NEXT-Schleifen anwendet, indem Sie die folgenden Aufgaben bearbeiten.

1. Schreiben Sie auf, was beim Lauf des folgenden Programms ausgedruckt wird.

```
10 S=0
20 FOR K=1 TO 4
30 S=S+K
40 NEXT K
50 PRINT S
RUN
```

....

2. Was wird durch das folgende Programm ausgedruckt?

```
10 P=1 : FOR K=1 TO 4 : P=P*K : NEXT K : PRINT P
RUN
```

....

3. Sehen Sie sich das folgende Programm genau an. Welcher der drei abgebildeten Ausdrucke wurde durch das Programm erzeugt? ......

....

```
10 N=1
20 FOR K=1 TO N
30 PRINT "*";
40 NEXT K
```

```
50 PRINT
60 N=N+1
70 IF N > 10 THEN END
80 GOTO 20
```

```
RUN 1
              *
             ***
            *****
           *******
          *********
         ***********
        *************
       ***************
```

```
RUN 2                          RUN 3
********                       *
 ********                      **
  ********                     ***
   ********                    ****
    ********                   *****
     ********                  ******
      ********                 *******
       ********                ********
                               *********
                               **********
```

4. Schreiben Sie ein Programm, durch das eine Tabelle für N, N^2 und N^3 ausgedruckt wird. Verwenden Sie INPUT-Statements, um anzugeben, welche Zahlen Sie in der Tabelle haben möchten. Ein Lauf sollte wie folgt aussehen:

```
RUN
ERSTE ZAHL? 40
LETZTE ZAHL? 45
```

N	N2	N3
40	1600	64000
41	1681	68921
42	1764	74088
43	1849	79507
44	1936	85184
45	2025	91125

....
....
....
....
....

. . . .

. . . .

5. Schreiben Sie auf, was beim Lauf des folgenden Programms gedruckt wird.

```
10 S=0
20 FOR K=1 TO 7 STEP 2
30 S=S+K
40 NEXT K
50 PRINT S
RUN
```

. . . .

6. Vervollständigen Sie bitte das folgende Programm, durch das eine Tabelle zur Vorhersage des Bevölkerungswachstums ausgedruckt wird, und zwar in vorgegebenen Abschnitten über eine bestimmte zeitliche Periode (Jahre). Die Formel für das Bevölkerungswachstum lautet:

$$Q = P(1 + R/100)^N$$

worin N die Anzahl der Jahre ist.
Die Bevölkerungszahl soll gerundet auf die nächste ganze Zahl ausgedruckt werden.

```
10 REM***DATENEINGABE, UEBERSCHRIFTEN
110 PRINT "ANFANGSBEVOELKERUNG"; : INPUT P
120 PRINT "WACHSTUMSRATE"; : INPUT R
130 PRINT "ANFANGSWERT VON N"; : INPUT A
140 PRINT "ENDWERT VON N"; : INPUT B
150 PRINT "SCHRITTWEITE"; : INPUT H
160 PRINT : PRINT "N", "BEVOELKERUNG" : PRINT
200 REM***BERECHNE UND DRUCKE DIE TABELLE
210 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
220 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
230 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
240 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
RUN
ANFANGSBEVOELKERUNG? 230
WACHSTUMSRATE? 1
ANFANGSWERT VON N? 0
ENDWERT VON N? 100
```

```
SCHRITTWEITE? 25
N              BEVOELKERUNG
0              230
25             295
50             378
75             485
100            622
RUN
ANFANGSBEVOELKERUNG? 205
WACHSTUMSRATE? 1
ANFANGSWERT VON N? 0
ENDWERT VON N? 100
SCHRITTWEITE? 10
N              BEVOELKERUNG
0              205
10             226
20             250
30             276
40             305
50             337
60             372
70             411
80             454
90             502
100            554
```

Die oben angegebene Anfangsbevölkerung in Millionen Einwohnern gilt für die USA im Jahre 1970. Die Ergebnisse sind ebenfalls in Millionen Einwohnern ausgedrückt, aberundet auf die nächste Million.

7. Schreiben Sie ein Programm zur Berechnung und zum Ausdrucken der Summe der ganzen Zahlen von 1 bis N, wober der Wert von N als Antwort auf ein INPUT-Statement eingegeben wird. Ein Lauf könnte dann wie folgt aussehen:

```
RUN
GEBEN SIE MIR EINE ZAHL (N), UND ICH WERDE IHNEN
DIE SUMME DER ZAHLEN VON 1 BIS N ERRECHNEN.
WIE LAUTET N? 3
DIE SUMME IST 6                (Da 1+2+3 = 6)
```

WIE LAUTET N? 5
DIE SUMME IST 15 (Da 1+2+3+4+5 = 15)
WIE LAUTET N? usw.

....
....
....
....
....
....
....
....
....
....
....

8. Sehen Sie sich noch einmal das einfache Computerspiel zum Erraten von Zahlen an, das in Kapitel 1, Abschnitt 9 besprochen wurde. Bauen Sie eine FOR-NEXT-Schleife so in dieses Programm ein, daß der Spieler nur 8 Versuche hat, um die Zahl zu erraten. Hat er sie nach 8 Versuchen noch nicht gefunden, lassen Sie den Computer eine passende Mitteilung ausdrucken, bevor er mit einem neuen Spiel beginnt.

....
....
....
....
....
....
....
....
....
....

9. Was wird bald auf dem Bildschirm erscheinen, wenn wir das folgende Programm speichern und dann ablaufen lassen? ....

....

```
100 FOR R=1 TO 23
110 FOR C=1 TO 37
120 S=INT(6*RND(1))+1
```

```
130 IF S=1 PRINT "*";
140 IF S>1 PRINT " ";
150 NEXT C
160 PRINT
170 NEXT R
180 GOTO 180
```

Antworten zum Eigentest

Die Zahlen in Klammern geben die Abschnitte an, in denen der jeweilige Stoff behandelt wurde.

1. 10
   Gedruckt wird die Summe der Werte von K, die durch das FOR-Statement defineirt werden. (K = 1,2,3 und 4). (Abschnitte 1 – 7)

2. 24
   Ausgedruckt wird das Produkt der Werte von K, die durch das FOR-Statement defineirt werden. (K = 1,2,3 und 4).
   (Abschnitte 1 – 6, 16)

3. RUN 3. Die FOR-NEXT-Schleife (Zeilen 20, 30, 40) veranlaßt den Computer, eine Reihe von N Sternen auszudrucken. Die Schleife wird ausgeführt für N = 1,2,3, . . . 10. (Abschnitte 1 – 6, 28)

4.
```
10 INPUT "ERSTE ZAHL "; : INPUT A
20 INPUT "LETZTE ZAHL "; : INPUT B
30 PRINT
40 PRINT "N", "N2", "N3"
50 FOR N=A TO B
60 PRINT N, N ↑ 2, N ↑ 3
70 NEXT N
```
   (Abschnitte 9, 21)

5. 16
   Auch durch dieses Programm werden wieder die durch das FOR-Statement definierten Werte von K addiert, die in diesem Fall aber 1,3,5 und 7 lauten. (Abschnitt 18)

6.
```
210 FOR N=A TO B STEP H
220 LET Q=P*(1+R/100) ↑ N
```

```
230 PRINT N, INT(Q+.5)
240 NEXT N
```

Zeile 230 druckt Q, gerundet auf die nächste ganze Zahl.
(Abschnitte 18 – 24)

7.
```
10 PRINT "GEBEN SIE MIR EINE ZAHL (N), UND ICH WERDE
   IHNEN"
20 PRINT "DIE SUMME DER ZAHLEN VON 1 BIS N
   BERECHNEN."
30 PRINT
40 PRINT "WIE LAUTET N";
50 INPUT N
55 LET S=0
60 FOR W=1 TO N
70 LET S=S+W
80 NEXT W
90 PRINT "DIE SUMME LAUTET";S
100 GOTO 30
```
(Abschnitt 2)

8.
```
100 REM***DIES IST EIN EINFACHES COMPUTER-SPIEL
110 LET X=INT(100*RND(1))+1
120 PRINT
130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND
    100."
140 PRINT "ERRATEN SIE MEINE ZAHL!!!"
150 FOR A=1 TO 8 : PRINT "IHR VERSUCH?" : GOTO 190
160 IF G < X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL" : GOTO 190
170 IF G > X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    KLEINEREN ZAHL" : GOTO 190
180 IF G=X THEN PRINT "SIE HABEN MEINE ZAHL ERRATEN."
    : GOTO 110
190 NEXT A : PRINT "SIE HABEN ZU OFT GERATEN. DIE
    ZAHL LAUTETE" ;X : GOTO 110
```
(Abschnitte 2, 4)

9. Auf dem Bildschirm werden die "Sterne,, aufgehen. Ungefähr 1/6 des Bildschirms wird Sterne enthalten, die restlichen 5/6 bleiben leer. Die Position der Sterne und Leerstellen wird willkürlich durch die Zeilen 120, 130 und 140 festgelegt.

NOTIZEN

KAPITEL 7

Indizierte Variable

In den Kapiteln 7 und 8 wollen wir Ihnen ein weiteres nützliches Hilfsmittel vorstellen, nämlich die indizierte Variable, wobei wir zunächst nur BASIC-Variable mit einem Index besprechen werden.

Sie werden dabei beispielsweise lernen, wie man die Stimmen einer Umfrage zählt und wie man Geldbeträge akkumuliert oder zählt und sie in verschiedene Gruppen einteilt. Außerdem werden Sie noch mehr praktische Erfahrungen im Umgang mit FOR-NEXT-Schleifen sammeln.

In diesem Kapitel werden viele neue Ideen für die Programmierung vorgestellt. Arbeiten Sie das Kapitel daher langsam und sorgfältig durch. Experimentieren Sie auch mit Ihrem Computer; Sie werden feststellen, daß Ihnen die Anwendung dieser Techniken eine wesentlich größere Flexibilität gibt. Wenn Sie dieses Kapitel beendet haben, können Sie . . .

- indizierte Variable mit einem Index erkennen und verwenden;
- indizierten Variablen Werte zuweisen;
- indizierte Variable verwenden, die auch im Index eine Variable haben;
- eindimensionale Felder (Arrays) verwenden, um den Wert indizierter Variablen zu speichern;
- das DIM-Statement verwenden, um dem Computer die maximale Größe der vom Programm verwendeten Arrays mitzuteilen.

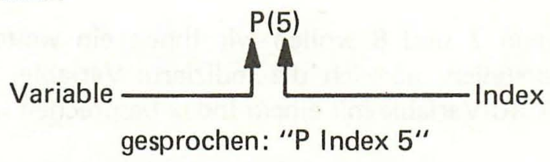
1. Das Konzept, das wir hier besprechen wollen, erfordert Ihre volle Aufmerksamkeit. Lesen Sie sich die Erklärungen langsam und sorgfältig durch, während wir uns allmählich in das geheimnisvolle Reich indizierter Variabler einarbeiten.

Bis jetzt haben wir nur einfache BASIC-Variable verwendet. Eine derartige Variable besteht aus einem beliebigen Buchstaben von A bis Z, gefolgt von einer einzelnen Zahl zwischen 0 und 9. Nachfolgend sehen

Sie einige einfache Variablen:

P R K P1 P2

Jetzt wollen wir einen neuen Variablentyp einführen, und zwar die indizierte Variable.



Eine indizierte Variable besteht aus einem beliebigen Buchstaben von A bis Z, gefolgt von einem in Klammern eingeschlossenen Buchstaben von A bis Z, gefolgt von einem in Klammern eingeschlossenen Index. P(3) ist eine indizierte Variable, P3 nicht.

Welche der folgenden Variablen sind indizierte?

X(1) X X1 C(23) D

X(1); C(23)

Anmerkung:
X, X1 und X(1) sind drei verschiedene Variable. Alle drei können somit gleichzeitig in einem Programm vorkommen. Das mag Sie zwar verwirren, nicht aber den Computer. Er wird sie ohne Schwierigkeiten unterscheiden.

2. Eine indizierte Variable bezeichnet, wie auch die einfachen Variablen, die wir bisher benutzt haben, einen Speicherplatz innerhalb des Computers. Sie können sich den Speicher als eine Schachtel vorstellen, in der man eine Zahl ablegen, also speichern kann.

Eine Anzahl mehrerer indizierter Variabler wird auch als "Array" bezeichnet. Die nachfolgend abgebildete Menge indizierter Variabler ist ein eindimensionales Array und wird auch als "Vektor" bezeichnet. In Kapitel 8 werden wir zweidimensionale Arrays besprechen.

P(0)	
P(1)	
P(2)	
P(3)	
P(4)	
P(5)	
P(6)	
P(7)	
P(8)	

Nehmen Sie einmal an, Sie seien der Computer und P(2) soll 36 sein. Nehmen Sie Ihren Bleistift und tragen Sie die Zahl 36 in die mit P(2) bezeichnete "Schachtel" in der obigen Zeichnung ein. Führen Sie in gleicher Weise die folgenden Zuweisungen aus: LET P(3) = 12 und LET P(7) = P(2) + P(3).

P(0)	
P(1)	
P(2)	36
P(3)	12
P(4)	
P(5)	
P(6)	
P(7)	48
P(8)	

3. Indizierte Variable können auch Variable als Indizes verwenden. Die indizierte Variable Y(J) beispielsweise hat die Variable J als Index.

Wenn J = 1 dann ist Y(J) = Y(1)
Wenn J = 2 denn ist Y(J) = Y(2)
Wenn J = 7 dann ist Y(J) = Y(7)

Nehmen wir einmal an, daß die Werte in den nachfolgend abgebildeten "Schachteln" den entsprechenden Variablen zugeordnet wurden. Be-

achten Sie, daß sowohl einfache als auch indizierte Variable vorkommen.

Y(1)	4	Z(1)	4.7	A	1
Y(2)	–3	Z(2)	9.2	B	2
Y(3)	5	X(1)	2	C	3
Y(4)	6	X(2)	3	D	4

Schreiben Sie für jede der nachfolgenden Variablen den Wert hin:

Y(1) = ...4...	A = ...1...	Y(A) = ...4...
Y(2) = ...-3...	B = ...2...	Y(B) = ...-3...
Y(C) = ...5...	X(A) = ...2...	X(B) = ...3...
Z(A) = ...4,7...	X(B) = ...3...	Y(D) = ...6...

— —

4	1	4
–3	2	–3
5	2	3
4.7	9.2	6

4. Bis jetzt haben wir nur einzelne Variable als Indizes verwendet. Der Index kann jedoch wesentlich komplexer sein. Hier sehen Sie zwei Beispiele, für die nochmals die Variablen und die Werte aus den "Schachteln" in Abschnitt 3 verwendet werden.

Y(A+1) = Y(1+1) = Y(2) = –3
Y(2*B) = Y(2*2) = Y(4) = 6

Beachten Sie, daß die Ausdrücke innerhalb der Indexklammern, unter Verwendung der gleichen BASIC-Regeln für die Arithmetik, wie in einem PRINT-Statement oder innerhalb der Klammern einer Funktion berechnet werden.

Vervollständigen Sie jetzt bitte die nachfolgenden Beispiele in der gleichen Weise, wie wir es Ihnen eben gezeigt haben. Schreiben Sie sowohl den berechneten Wert für den Index als auch den der indizierten Variablen mit diesem Index zugewiesenen Wert hin.

(Beziehen Sie sich dabei auf die Zahlen in Abschnitt 3.)

Y(A+2) = ...

Y(2*A–1) = ...

Y(A+B) = ...

Y(B*C–D) = ...

Y(A+3) = ...

Y(D–3) = ...

Y(D–C+A) = ...

Y((C+D)–(A+B)) = ...

– –

Y(A+2) = Y(1+2) = Y(3) = 5

Y(2*A–1) = Y(2*1–1) = Y(2–1) = Y(1) = 4

Y(A+B) = Y(1+2) = Y(3) = 5

Y(B*C–D) = Y(2*3–4) = Y(6–4) = Y(2) = –3

Y(A+3) = Y(1+3) = Y(4) = 6

Y(D–3) = Y(4–3) = Y(1) = 4

Y(D–C+A) = Y(4 – 3 + 1) = Y(1+1) = Y(2) = –3

Y((C+D)–(A+B)) =Y((3+4)–(1+2)) = Y(7–3) = Y(4) = 6

5. Sie werden sich sicher noch daran erinnern, daß wir dem Computer mitteilen mußten, wieviel Platz er in seinem Speicher für die Strings reservieren soll, die den String-Variablen in Ihrem Programm zugewiesen werden. In gleicher Weise müssen Sie auch indizierte Variable für die maximale Anzahl von Werten dimensionieren, die einer einzelnen indizierten Variablen zugewiesen werden sollen. Das heißt, Sie müssen dem Computer durch ein DIM-Statement den größten Index mitteilen, den er für eine indizierte Variable zulassen soll. DIM ist die Abkürzung für die "Dimension" eines Arrays indizierter Variabler. Die DIM-Statements müssen in Ihrem Programm so angeordnet werden, daß ihre Ausführung erfolgt, bevor die indizierten Variablen tatsächlich im Programm benutzt werden. Andernfalls erfolgt eine Fehlermeldung und die Programmausführung wird angehalten. Wie man ein Array dimensioniert ist anschließend zu sehen. Beachten Sie die Ähnlichkeit mit der Dimensionierung einer String-Variablen.

```
100 DIM X(100)
```

die Variable, für die Platz reserviert werden soll → X

(100) → maximal zulässiger Index

Das obige DIM-Statement spezifiziert eine indizierte Variable die als höchsten Index .... haben kann.

100

6. Nehmen wir einmal an, wir möchten festlegen, daß der maximale Index 50 ist. Bitte schreiben Sie dafür ein DIM-Statement.

105 ....

---

105 DIM X(50)

7. Wie können indizierte Variable nun zu einr Vereinfachung und Verbesserung der Programmierung in BASIC beitragen? Eine sehr häufige Anwendung indizierter Variablen besteht beispielsweise darin, eine Liste von Zahlen, die über INPUT- oder READ-Statements eingegeben wurden, zu speichern. Das kann mit einer FOR-NEXT-Schleife erfolgen. Die Kontrollvariable läßt sich auch als Variable für den Index einer indizierten Variablen verwenden, was zu Folge hat, daß der Index bei jedem Schleifendurchlauf um 1 erhöht wird. Um dies zu verdeutlichen, verwenden wir noch einmal das bereits bekannte Programm "DIE TEUERSTE ADDIERMASCHINE".

```
100 REM***DIE TEUERSTE ADDIERMASCHINE
105 REM *** DER WELT
110 DIM X(10)
120 PRINT "WIEVIELE ZAHLEN";
130 INPUT N
140 PRINT
150 FOR K=1 TO N
160 PRINT "X=";
170 INPUT X
180 X(K)=X
190 NEXT K
200 F=0
210 FOR K=1 TO N
220 LET T=T+X(K)
230 NEXT K
240 PRINT "DIE SUMME IST"; T
```

```
RUN
WIEVIELE ZAHLEN? 5
X=?37
X=?23
X=?46
X=?78
X=?59
DIE SUMME IST 243
```

Im obigen Lauf ist N gleich 5. Daher werden vom Operator fünf Zahlen eingegeben und in X(1) bis .... gespeichert.

X(5)

8. Sehen Sie sich vor allem die Zeilen 170 und 180 im vorigen Abschnitt noch einmal an. Zuerst wird der durch INPUT eingegebene Wert der einfachen, numerischen Variablen X zugeordnet. Dann wird der gleiche Wert der indizierten Variablen X(K) zugewiesen und zwar durch X(K) = X.

Nehmen wir an, der Computer arbeitet das Programm in Abschnitt 7 ab und hat gerade die FOR-NEXT-Schleife in den Zeilen 150 bis 190 abgeschlossen. Die vom Operator eingegebenen Zahlen werden jetzt folgendermaßen gespeichert.

N	5
X(1)	37
X(2)	23
X(3)	46
X(4)	78
X(5)	59

Der Computer ist bereit, mit Zeile 200 fortzufahren. Dieses Statement initialisiert die Variable T, d.h. es weist T den ersten Wert zu. Schreiben Sie den Wert für T hin, nachdem die Zeile 200 ausgeführt wurde.

T [   ]

T [0]

9. Als nächstes wird der Computer die FOR-NEXT-Schleife in den Zeilen 210 bis 230 ausführen. Wie oft wird diese Schleife durchlaufen? ......

---

fünfmal, da Zeile 210 sagt: FOR K = 1 TO N und N gleich 5 ist.

10. Die FOR-NEXT-Schleife in den Zeilen 210 – 230 wird fünfmal ausgeführt, zuerst für K = 1, dann für K = 2, K = 3 und K = 4, schließlich für K = 5. Sehen wir uns einmal Zeile 220 an, in der K als Index benutzt wird.

```
220 LET T=T+X(K)
```

Dieses Statement weist den Computer an, den Wert von X(K) zum alten Wert von T zu addieren und dann das Ergebnis als neuen Wert von T nehmen.

Wie lautet der Wert von T, nach Ausführung von Zeile 220, für K = 1? ... , für K = 2? ...., für K = 3? ...., für K = 4? ...., für K = 5? ....

37, 60, 106, 184, 243

11. Lassen Sie uns einmal die bereits bekannte "teuerste Addiermaschine der Welt" dazu verwenden, die ganzen Zahlen von 1 bis 12 zu addieren.

```
RUN
WIEVIELE ZAHLEN? 12
X =? 1
X= ? 2
X= ? 3
X= ? 4
X= ? 5
X= ? 6
```

```
X= ? 7
X= ? 8
X= ? 9
X= ?10
X= ? 11
ERROR— AT LINE 180
```

Diesmal erhalten wir eine Fehlermeldung, die uns sagt, daß wir einen falschen Index verwendet haben. Welches war der größte Index für X(K), den der Computer noch akzeptierte, vevor er eine Fehlermeldung ausgab? . . . .

— —

10

12. Ändern Sie die Zeile 110 durch folgendes Statement so ab, daß wir die Möglichkeit haben, mehr Zahlen einzugeben:

```
110 DIM X(100)
```

Lassen Sie jetzt das Programm nochmals ablaufen, und zwar unter Verwendung der 12 Zahlen, die uns zuvor Schwierigkeiten bereitet haben.

```
RUN
WIEVIELE ZAHLEN? 12
X =? 1
X =? 2
X =? 3
X =? 4
X =? 5
X =? 6
X =? 7
X =? 8
X =? 9
X =? 10
X =? 11
X =? 12
DIE SUMME IST 78
```

Die Summe wievieler Zahlen maximal kann das Programm jetzt berechnen? . . . .

---

100. Wenn 100 Zahlen eingegeben werden, erfolgt Ihre Speicherung in X(1) bis X(100). Diese Grenze ist durch das DIM-Statement in Zeile 110 festgelegt. Wir können das Programm natürlich dazu verwenden, um die Summe von weniger als 100 Zahlen zu berechnen.

13. Die Dimensionen eines DIM-Statements können auch durch eine Variable zugewiesen werden. Sehen Sie sich dazu die folgenden Beispiele an:

DIM X(N)　　　　DIMA(B)　　　　DIM D(Y+2)

Die dürfen ein Array während eines Programmlaufs jedoch nur einmal dimensionieren. Der Computer wird Ihnen sonst eine Fehlermitteilung ausgeben und die Programmausführung stoppen, wenn er für das gleiche Array zum zweitenmal zu einem DIM-Statement gelangt und während des gleichen Laufs versucht, das Array umzudimensionieren.

Sehen Sie sich das anschließende Programm-Segment und den Beginn des Laufs an. Wir haben Zeile 110 gelöscht und das DIM-Statement an den Anfang von Zeile 150 gestellt.

```
.
.
.
120 PRINT "WIEVIELE ZAHLEN";
130 INPUT N
140 PRINT
150 DIM X(N) : FOR K=1 TO N

:
:
RUN
WIEVIELE ZAHLEN: 8
```

Wie groß wird die dimensionierte Größe des Arrays während dieses Programmlaufs sein? . . . .

---

8 (das ist der Wert, der N durch das INPUT-Statement zugewiesen wurde)

14. Wo könnte DIM X(N) in dem in Abschnitt 13 gezeigten Programm-Segment noch angeordnet werden? ....
....

— — — — — — — — — — — — — — — — — — — —

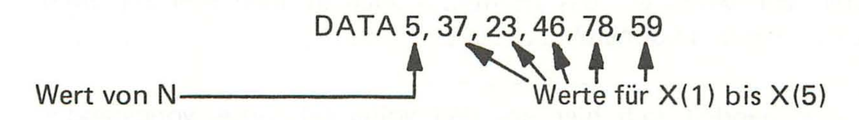
In Zeile 130, hinter dem INPUT-Statement, oder in Zeile 140 vor oder nach PRINT. Beachten Sie aber, daß das DIM-Statement an einer Stelle des Programms angeordnet werden muß, an der bereits der INPUT-Wert eingegeben wurde, aber vor der FOR-NEXT-Schleife, sodaß das Array X(K) während des gleichen Laufs nur einmal dimensioniert wird. Das Statement DIM X(N) könnte auch in einer Zeile für sich allein stehen, unter Verwendung einer Zeilennummer zwischen 130 und 140 oder zwischen 140 und 150.

15. Sehen Sie sich noch einmal die Zeilen der ersten FOR-NEXT-Schleife (150 – 190) des Programms in Abschnitt 7 an. Der erste Wert, der K zugewiesen wird, ist 1, sodaß X(1) die erste indizierte Variable ist. Da das Array mit X(0) beginnen könnte, kann ein durch DIM X(100) dimensioniertes Array wieviele Werte aufnehmen? ....

— — — — — — — — — — — — — — — — — — — —

101
FOR K=0 TO N

16. Statt eines INPUT-Statements zur Eingabe der Werte für X(1), X(2) usw. können wir auch READ- und DATA-Statements benutzen. Den Wert für N sowie die Werte für X(1) bis X(N) setzen wir dann folgendermaßen in ein DATA-Statement ein:




```
100 REM***DIE TEUERSTE ADDIERMASCHINE DER WELT
110 REM X(100)
200 REM***LESEN VON N UND X(1) BIS X(N)
210 READ N
220 FOR J=1 TO N
230 READ X : X(J)=X
240 NEXT J
```

```
300 REM***DRUCKEN DES WERTES FÜR N UND DES X-ARRAYS
310 PRINT "N="; N
320 PRINT "X(1) BIS X(";N;") LAUTEN:"
330 FOR K=1 TO N
340 PRINT " "; X(K);
350 NEXT K
360 PRINT
400 REM***SUMME DER WERTE DES X-ARRAYS
410 T=0
420 FOR L=1 TO N
430 T=T+X(L)
440 NEXT L
500 REM***DRUCKEN DER SUMME; NEUER START
510 PRINT "DIE SUMME IST "; T
520 PRINT
530 GOTO 210
900 REM***HIER SIND ZWEI DATENSAETZE
910 DATA 5,37,23,46,78,59
920 DATA 12,1,2,3,4,5,6,7,8,9,10,11,12
```

In der ersten FOR-NEXT-Schleife (Zeilen 200 – 240) haben wir J als Index benutzt. Diese Wahl erfolgte völlig willkürlich.
Welchen Index verwendeten wir in den Zeilen 330 bis 350? . . . .
und in den Zeilen 420 bis 440? . . . .

– – – – – – – – – – – – – – – – – – – –

K; L

17. Hätten wir, wenn wir das wollten, J auch in allen drei Schleifen als Index verwenden können? . . . .

– –

Ja, denn es handelt sich hier um drei völlig getrennte, voneinander verschiedene FOR-NEXT-Schleifen. Wir hätten jede Variable als Index verwenden können, außer N, T oder X.

18. Die folgenden Fragen beziehen sich noch auf das Programm in Abschnitt 16.

a) Welches Statement weist, beim zweiten Datensatz, den ersten Wert im DATA-Statement einer Variablen zu? . . . .

b) Welche Statements weisen den Rest der Daten einer indizierten Variablen zu? ....

c) Welches Statement druckt die im Array gespeicherten Werte, statt der Werte aus dem DATA-Statement? ....

d) Welches Statement addiert alle im Array gespeicherten Werte auf?

....

– – – – – – – – – – – – – – – – – – – –

a) 210 READ N
b) 230 READ X : X(J)=X
c) 340 PRINT X(K);
d) 430 T=T+X(L)

Anmerkung:
Zum besseren Verständnis von Zeile 230 sehen Sie sich bitte noch einmal Abschnitt 8 an.

19. Nehmen wir an, wir lassen das Programm in Abschnitt 19 ablaufen. Schreiben Sie bitte auf, wie ein RUN aussehen wird (Hinweis: überprüfen Sie alle PRINT-Statements).
RUN

....
....
....
....
....
....
....
....
....
....

– –

```
N=5
X(1) BIS X(5) LAUTEN:
 1  2  3  4  5  6  7  8  9  10  11  12
DIE SUMME IST 78

ERROR— 6 AT LINE 210
```

```
N=12
X(1) BIS X(12) LAUTEN:
  1   2   3   4   5   6   7   8   9   10   11   12
DIE SUMME IST 78

ERROR— 6 AT LINE 210
```

20. Damit Sie indizierte Variable noch besser verstehen und beim Schreiben Ihrer Programme anwenden können, wollen wir uns hier noch einmal etwas näher ansehen, wie der Computer arbeitet, wenn er es mit indizierten Variablen zu tun hat.

Bitte tragen Sie für das folgende Segment eines Computer-Programms in den leeren Feldern die Werte für D(G) ein, nachdem die FOR-NEXT-Schleife ausgeführt wurde.

```
10 DIM D(3)
20 FOR G=1 TO 3
30 D(G)=2*G–1
40 NEXT G
```

D(1) [   ]   D(2) [   ]   D(3) [   ]

D(1)	1	Für G=1 ist 2*G–1=2*1–1=2–1=1
D(2)	3	Für G=2 ist 2*G–1=2*2–1=4–1=3
D(3)	5	Für G=3 ist 2*G–1=2*3–1=6–1=5

21. Tragen Sie bitte für die folgende FOR-NEXT-Schleife in den Feldern R(1) bis R(4) die jeweiligen Werte ein, nachdem die Schleife ausgeführt wurde. (Denken Sie daran, daß R und (R) verschiedenen Variable sind.)

```
10 DIM R(10)
20 FOR R=1 TO 4
30 R(R)=R+2
40 NEXT R
```

R(1) [ ]   R(2) [ ]   R(3) [ ]   R(4) [ ]

R(1)	3		Für R=1 ist R+2=1+2=3
R(2)	4		Für R=2 ist R+2=2+2=4
R(3)	5		Für R=3 ist R+2=3+2=5
R(4)	6		Für R=4 ist R+2=4+2=6

22. Lassen Sie uns dies noch an einem weiteren Beispiel üben.
Tragen Sie bitte in die freien Felder die jeweiligen Werte nach der Ausführung der FOR-NEXT-Schleife ein.

```
10 DIM P(10)
20 FOR N=1 TO 6
30 P(N)=2*N
40 NEXT P
```

P(1) [ ]   P(2) [ ]   P(3) [ ]

P(4) [ ]   P(5) [ ]   P(6) [ ]

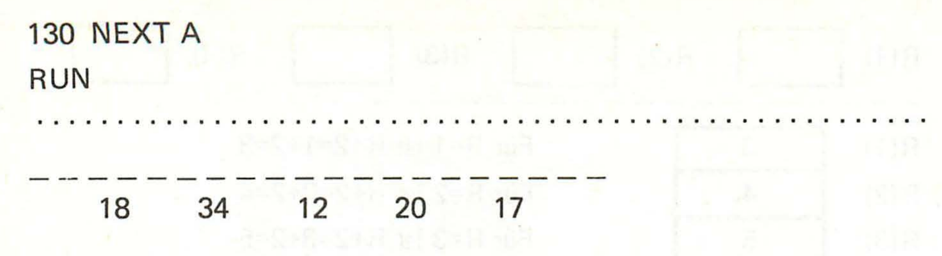
P(1)	2	P(2)	4	P(3)	6
P(4)	8	P(5)	10	P(6)	12

23. Nehmen wir an, daß in C(1) bis C(5) folgende Zahlen gespeichert sind:

C(1)	18	C(2)	34	C(3)	12
C(4)	20	C(5)	17		

Was wird ausgedruckt, wenn die folgende FOR-NEXT-Schleife ausgeführt wird?

```
100 DIM C(10)
110 FOR A=1 TO 5
120 PRINT " "; C(A);
```



```
130 NEXT A
RUN
```

....

- -

```
      18    34    12    20    17
```

24. Nehmen wir an, daß die Zahlen in C(1) bis C(5) gespeichert sind, wie es Abschnitt 23 zeigt. Was wird dann bei Ausführung der folgenden Schleife ausgedruckt?

```
100 DIM C(10)
110 FOR A=5 TO 1 STEP -1
120 PRINT " "; C(A);
130 NEXT A
RUN
```

....

- -

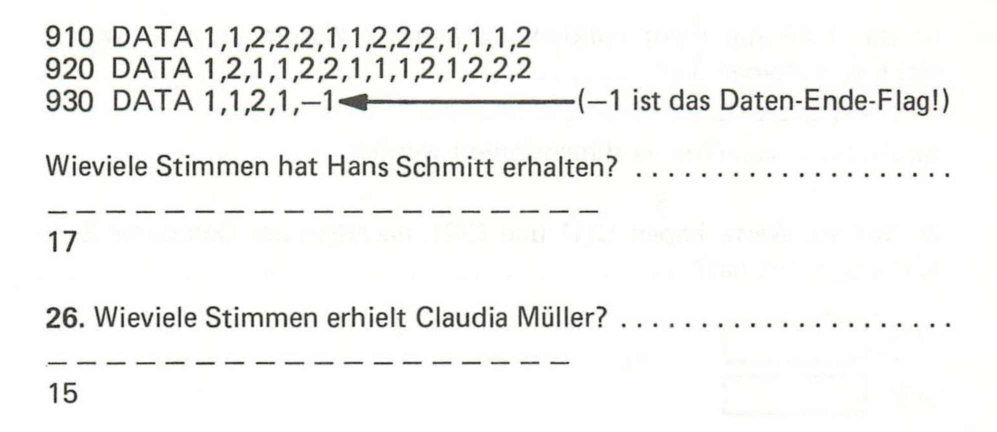
```
      17    20    34    18
```

Die Zahlen werden in umgekehrter Reihenfolge ausgedruckt. Wenn Sie das nicht herausbekommen haben, sehen Sie sich noch einmal die Abschnitte 18 und 19 in Kapitel 6 an.

25. Nehmen wir einmal an, eine Wahl steht bevor und Sie haben, unter Verwendung der nachfolgenden Fragen, bei Ihren Freunden eine Umfrage durchgeführt:

> Für wen werden Sie in der kommenden Wahl stimmen? Kreisen Sie die Ziffer links vom jeweiligen Kandidaten ein.
>
> 1. HANS SCHMITT
> 2. CLAUDIA MUELLER

Wir wollen jetzt ein Programm zum Zählen der Stimmen schreiben, die jeder Kandidat bei der Umfrage erhalten hat. Sie haben 35 Antworten auf Ihre befragung vorliegen, von denen jede entweder eine "1" oder eine "2" ist. Tragen Sie die Abstimmergebnisse zuerst in DATA-Statements ein.



```
910 DATA 1,1,2,2,2,1,1,2,2,2,1,1,1,2
920 DATA 1,2,1,1,2,2,1,1,1,2,1,2,2,2
930 DATA 1,1,2,1,–1◄——————(–1 ist das Daten-Ende-Flag!)
```

Wieviele Stimmen hat Hans Schmitt erhalten? ....

17

26. Wieviele Stimmen erhielt Claudia Müller? ....

15

27. Um die beiden letzten Fragen zu beantworten, haben Sie sicherlich zuerst die "Einsen" in den DATA-Statements gezählt, um herauszufinden, wieviele Stimmen Hans Schmitt erhalten hat. Anschließend haben Sie die "Zweien"gezählt, um die Stimmen für Claudia Müller zu ermitteln.

Ihr Computer kann Ihnen diese Arbeit jedoch abnehmen und die Stimmen zählen, indem er indizierte Variable verwendet, um die Gesamtsummen der "Einsen" und "Zweien" in den DATA-Statements zu ermitteln. Wenn er zum "Daten-Ende-Flag" gelangt, hört er mit dem Zählen auf und druckt die Ergebnisse. Hier ist das Programm zum Zählen der Stimmen.

```
100 REM***STIMMEN-ZAEHLPROGRAMM
110 REM***INITIALISIERUNG
120 DIM C(2) : C(1)=0 : C(2)=0
200 REM***LIES UND ZAEHLE DIE STIMMEN
210 READ V
220 IF V=–1 THEN 310
230 C(V)=C(V)+1
240 GOTO 210
300 REM***DRUCKE DIE ERGEBNISSE
310 PRINT "HANS SCHMITT:"; C(1)
320 PRINT "CLAUDIA MUELLER:"; C(2)
910 DATA 1,1,2,2,2,1,1,2,2,2,1,1,1,2
920 DATA 1,2,1,1,2,2,1,1,1,2,1,2,2,2
930 DATA 1,1,2,1,–1
RUN
HANS SCHMITT: 17
CLAUDIA MUELLER: 15
```

Ist das DIM-Statement wirklich notwendig? Warum ja, oder warum nicht, was meinen Sie? ....

Ja; Arrays müssen immer dimensioniert werden.

28. Welche Werte haben C(1) und C(2), nachdem der Computer Zeile 120 ausgeführt hat? ....

C(1) [ ]

C(2) [ ]

Beide haben den Wert 0. Dabei handelt es sich um die Anfangswerte, bevor mit dem Zählen der Stimmen begonnen wird. Wir nennen diesen Vorgang Initialisierung, wie es auch durch das REM-Statement in Zeile 110 angedeutet wird.

29. Wenn Ihre Programme so lang werden, daß sie anfangen, die Speicher-Kapatzität Ihres Computers zu übersteigen, dann lassen Sie die REM-Statements weg, um Platz zu gewinnen. Dabei sparen Sie auch noch Zeit für das Abschreiben, wenn Sie die Programme nicht für spätere Verwendung speichern wollen.

Welche Statements könnte man im Programm aus Abschnitt 27 weglassen, ohne daß die einwandfreie Arbeitsweise beeinträchtigt wird?

....

100, 110, 200, 300

30. Sehen Sie sich jetzt noch einmal das Statement mit der Zeilennummer 230 aus dem vorigen Programm an:

230 LET C(V)=C(V)+1

Bis auf die Tatsache, daß bei diesem Statement indizierte Variable verwendet werden, ist es äquivalent zu einem Statement, das bereits in früheren Programmen, unter anderem zum Zählen, verwendet wurde:

LET N=N+1

Achten Sie darauf, wie der variable Index von C dazu verwendet wird, um zu bestimmen, ob der Wert von C(1) oder der Stand von C(2) um eins erhöht wird. Da V nur zwei Werte haben kann, nämlich entweder 1 oder 2, erfüllt Zeile 230 tatsächlich einen doppelten Zweck. Abhängig vom jeweiligen Wert ist dieses Statement äquivalent zu.

LET C(1)=C(1)+1 oder LET C(2)=C(2)+1

Welche Werte hat der Computer für C(1) und C(2) beim Ablauf des Programms, nach dem Lesen und der Verarbeitung der ersten Stimme gespeichert? (Das heißt, die Zeilen 210 bis 230 wurden für die erste Stimme im ersten DATA-Statement ausgeführt.)

C(1) [ ] C(2) [ ]

Welche Werte werden für C(1) und C(2) gespeichert, nachdem die zweite Stimme gelesen und ausgewertet wurde.

C(1) [ ] C(2) [ ]

Welche Werte sind in C(1) und C(2) nach dem Lesen und der Verarbeitung der dritten Stimme gespeichert?

C(1) [ ] C(2) [ ]

C(1) [1] C(2) [0]

C(1) [2] C(2) [0]

C(1) [2] C(2) [1]

31. In dem Programm, das wir diskutiert haben (Abschnitt 27), prüft Zeile 220 mit folgendem Statement, ob das Daten-Ende-Flag vorliegt:

IF V=–1 THEN 310

Wenn wir die Zeilen 220 und 230 vertauschen, sieht das Programm folgendermaßen aus:

```
210 READ V
220 LET C(V)=C(V)+1
230 IF V=-1 THEN 310
```

240 GOTO 210

a) Wie lautet jetzt der letzte Wert, der V aus dem DATA-Statement zugewiesen wird? ....

b) Wenn der Computer diesen Wert für V in Zeile 220 verwenden würde, wie lautete dann die indizierte Variable? ....

c) Welche Fehlermeldung würde unser Computer ausgeben, da negative Indizes nicht erlaubt sind? ....

---

a) –1

b) C(–1)

c) ERROR– falscher Index. Moral: Achten Sie darauf, wo Sie die Abfrage für das Daten-Ende-Flag einfügen. Die beste Stelle ist gewöhnlich direkt hinter dem READ- oder INPUT-Statement, in dem das Flag auftauchen kann.

32. Nehmen wir einmal an, es wird folgende Abstimmung durchgeführt:

Für welchen Kandidaten würden Sie bei der nächsten Wahl stimmen? Kreisen Sie die Zahl links von Ihrem Kanditaten ein. 1. Hans Schmitt 2. Claudia Müller 3. Keine Meinung

Die Ergebnisse dieser Umfrage sehen Sie hier:

2,2,2,1,2,1,1,2,1,1,3,2,1,3
2,1,1,3,1,3,2,2,1,1,3,2,1,3
1,1,2,1,2,1,1

Modifizieren Sie das Stimmen-Zählprogramm aus Abschnitt 27 so, daß sich diese Daten damit verarbeiten lassen. Sie müssen u. a. eine Zeile einfügen, um C(3) auf 0 zu setzen, außerdem ein PRINT-Statement, um die Anzahl der Stimmen mit "Keine Meinung". auszudrucken. Desgleichen müssen Sie die DATA-Statements für die neuen Daten, sowie das DIM-Statement ändern. Hier sehen Sie, wie der Lauf aussehen sollte.

```
RUN
HANS SCHMITT: 17
CLAUDIA MUELLER: 12
KEINE MEINUNG: 6
```

Schreiben Sie hier Ihr Programm hin:

```
100 REM***STIMMEN-ZAEHLPROGRAMM
110 REM***INTITIALISIERUNG
120 DIM C(3)
130 C(1)=0 : C(2)=0 : C(3)=0
200 REM***LIES UND ZAEHLE DIE STIMMEN
210 READ V
220 IF V=–1 THEN 310
230 C(V)=C(V)+1
240 GOTO 210
300 REM***DRUCKE DIE ERGEBNISSE
310 PRINT "HANS SCHMITT:"; C(1)
320 PRINT "CLAUDIA MUELLER:"; C(2)
330 PRINT "KEINE MEINUNG:"; C(3)
910 DATA 2,2,2,1,2,1,1,2,1,1,3,2,1,3
920 DATA 2,1,1,3,1,3,2,2,1,1,3,2,1,3
930 DATA 1,1,2,1,2,1,1,–1 ←——— (haben Sie an das Flag dedacht?)
```

33. Nehmen wir an, wir führen eine Umfrage mit vier möglichen Antworten, oder gar fünf oder sechs durch. Statt jeweils ein neues Programm zu schreiben, wollen wir eins schreiben mit dem sich die Stimmen einer Berfragung mit N möglichen Antworten zählen lassen. Der Wert von N muß dann in einem DATA-Statement vor den tatsächlichen Antworten bzw. Stimmen stehen. Die Daten für die Befragung in Abschnitt 25 würden dann beispielsweise folgendermaßen aussehen.

```
900 REM***DIE DATEN
901 REM***DER ERSTE WERT IST DIE ZAHL DER
902 REM***MOEGLICHEN ANTWORTEN
910 DATA 2
920 DATA 1,1,2,2,2,1,1,2,2,2,1,1,1,2
930 DATA 1,2,1,1,2,2,1,1,1,2,1,2,2,2
940 DATA 1,1,2,1,–1
```

In Zeile 910 steht der Wert für N. In diesem Fall ist N=2 und die möglichen Antworten lauten 1 und 2. Wie müßten die Ergebnisse der Befragung in Abschnitt 32 in DATA-Statements eingefügt werden?

```
900 REM***DIE DATEN
901 REM***DER ERSTE WERT IST DIE ZAHL DER
902 REM***MOEGLICHEN ANTWORTEN
910 DATA ....................................
919 REM***DIE STIMMEN UND DAS FLAG (–1)
920 .........................................
930 .........................................
940 .........................................
```

```
910 DATA 3
920 DATA 2,2,2,1,2,1,1,2,1,1,3,2,1,3
930 DATA 2,1,1,3,1,3,2,2,1,1,3,2,1,3
940 DATA 1,1,2,1,2,1,1,--1
```

Diesmal ist N=3 (Zeile 910) und die möglichen Entscheidungen sind 1, 2 oder 3.

34. Jetzt sind Sie daran. Schreiben Sie ein Programm, um die Stimmen bei einer Befragung mit N verschiedenen Antworten zu lesen und zu zählen. N soll kleiner oder gleich 20 sein. In diesem Programm müssen folgende Verarbeitungsschritte vorgesehen werden:

1. Dimensionierung des maximal möglichen Index von C. Erinnern Sie sich daran, daß N kleiner oder gleich 20 ist.
2. Lesen des Wertes von N.
3. C(1) bis C(N) auf 0 setzen. (Verwenden Sie dazu eine FOR-NEXT-Schleife.)
4. Lesen und Zählen der abgegebenen Stimmen, bis ein Flag gelesen wird.
5. Ausdrucken der Ergebnisse, so wie es das folgende Beispiel zeigt:

Beispiel: N = 2

```
RUN
ANTWORT 1 : 17
ANTWORT 2 : 15
```

Beispiel: N = 5

```
RUN
ANTWORT 1 : 12
ANTWORT 2 : 7
ANTWORT 3 : 9
ANTWORT 4 : 9
ANTWORT 5 : 10
```

Die beiden abgebildeten Programmläufe wurden mit nachfolgenden Datensätzen erzeugt:

```
910 DATA 2
920 DATA 1,1,2,2,2,1,1,2,2,2,1,1,1,2
930 DATA 1,2,1,1,2,2,1,1,1,2,1,2,2,2
940 DATA 1,1,2,1,–1
```

```
910 DATA 5
920 DATA 4,3,4,2,4,1,1,5,5,3,5,4,5,1
930 DATA 3,2,5,5,4,4,5,1,2,3,3,3,5,2
940 DATA 2,3,1,5,4,1,1,1,2,3,1,4,1,5
950 DATA 1,2,3,4,1,–1
```

Schreiben Sie jetzt Ihr eigenes Programm. (Die DATA-Statements brauchen Sie nicht mehr vorzusehen.)

. . . .

. . . .

. . . .

. . . .

\- - - - - - - - - - - - - - - - - -

```
100 REM***STIMMEN-ZAEHLPROGRAMM
110 REM***INITIALISIERUNG
120 DIM C(20) : READ N : FOR K=1 TO N : C(K)=0 : NEXT K
200 REM***LESEN UND ZAHLEN DER STIMMEN
210 READ V : IF V <> -1 THEN LET C(V)=C(V)+1 : GOTO 210
300 REM***AUSDRUCKEN DER ERGEBNISSE
310 FOR K=1 TO N : PRINT "ANTWORT"; K; ";"; C(K) : NEXT K
```

35. In den Abschnitten 25 – 34 war das Zählen von Stimmen das zentrale Problem. Dazu wurde bei jedem Schleifendurchlauf, in dem Daten gelesen werden, eine 1 zu einem Array-Element hinzuaddiert (LET C(V) + 1). Nahezu jedes Zählproblem kann auf eine Weise behandelt werden, die mit den Lösungen in Abschnitt 34 vergleichbar ist.

Eine weitere, ähnliche Anwendung einfacher, eindimensionaler Arrays besteht darin, Gegenstände oder auch Geld zu zählen.

Um dies an einem praktischen Beispiel zu erproben, wollen wir einmal annehmen, Ihr Computerclub möchte ein neues Terminal kaufen, das als Bausatz 1.200,– DM kosten soll. Die Frage ist nur, woher soll das Geld dafür kommen? Da die meisten Mitglieder noch sehr jung sind, oder in die Schule gehen, wird beschlossen, Schockoladenriegel an Mitschüler oder Freunde für 1 DM pro Stück zu verkaufen, die der Club für 55 Pfg. einkauft. Ein ganz hübscher Profit also. Sie sollen die Buchführung übernehmen und geben dazu jedem Clubmitglied eine Nummer. Immer wenn eins der Mitglieder vorbeikommt, um neue "Ware" abzuholen, schreiben Sie seine Nummer und die Anzahl der Schokoladenriegel auf. Das Geld wird dann später abgerechnet.

Hier sind die Indentifikations-Nummern der Club-Mitglieder.

1. Jerry	5. Karl
2. Bobby	6. Mimi
3. Mary	7. Doug

4. Danny 8. Scott

Nehmen wir an, Danny holt 6 Schokoladenriegel. Sie notieren dann 4,6. Die 4 ist Dannys Identifikationsnummer für den Computer, 6 ist die Anzahl der Schikoladenriegel. Wenn Doug 12 Riegel mitnimmt, nottieren Sie ..... Holt Mary 6 Riegel, so lautet Ihre Aufzeichnung ....

- - - - - - - - - - - - - - - - - -

7,12 3,6

36. Nach einigen Wochen haben Sie bereits eine ganze Reihe derartiger Notierungen vorliegen, so daß es Zeit wird, einmal aufzuaddieren, wieviel Geld bisher zusammengekommen ist. Die in Ihren Aufzeichnungen enthaltenen Informationen werden dazu in DATA-Statements eingesetzt. Sie könnten die Daten auch unter Verwendung von INPUT-Statements eingeben, aber das würde viel zu lange dauern.

```
900 REM***DATENPAARE: INDENTIFIKATIONS-NUMMER UND
901 REM***MENGE
910 DATA 4,6,7,12,3,6,1,8,4,5,5,3,8,20
```

Hier sehen Sie, daß Danny nochmals 5 Riegel geholt hat.

```
920 DATA 2,4,3,8,6,6,5,10,7,12,8,4,1,3
```

Mary hat weitere 8 geholt.

```
930 DATA -1, -1
```

a) Aus wieviel Zahlen besteht jeder Datensatz? ....
b) DATA 6,6 bedeutet: ....
c) Aus wieviel Zahlen besteht das End-of-Data-Flag? ....
Warum? ....

- -

a) zwei
b) Mimi nahm 6 Riegel
c) Zwei. Jedes Daten-Element besthet aus zwei Zahlen. Das heißt, daß mit jedem READ-Statement zwei Zahlen gelesen werden. Würde daher für die beiden Variablen im READ-Statement nur ein Flag vorgesehen werden, erhielte man eine Fehler-Meldung.

37. Während Sie noch dabei sind, die Daten einzugeben, kommt Bobby vorbei und holt nochmals sechs Riegel. Schreiben Sie auf, wie man diese Daten hinzufügen kann, ohne eins der bereits vorhandenen DATA-Statements umzuändern. Verwenden Sie die höchstmögliche Zeilennummer, sodaß die Daten immer noch vor den Daten-Ende-Flags liegen, die als letzte Werte vom Programm gelesen werden. ....

– –

929 DATA 2,6

Die Zeilennummer darf nicht größer als 929 sein, da sonst die in Zeile 930 vorgesehenen Flags nicht als letzte Daten im Programm gelesen werden. Es ist daher von großer Bedeutung, an welcher Stelle ein DATA-Statement in ein bereits vorhandenes Programm eingefügt wird!

38. Wir brauchen ein Array mit 8 Elementen. Das bedeutet, daß wir eine indizierte Variable mit Indizes von 1 bis 8 verwenden müssen, um dadurch die 8 Clubmitglieder zu repräsentieren. Der zu jedem Element des Arrays, also zu den einzelnen indizierten Variablen, hinzuaddierte Wert ist die Anzahl der Schokoladenriegel, die jedes Miglied verkauft hat. Zuerst muß das Array jedoch initialisiert werden. Schreiben Sie dazu eine Zeile mit mehreren Statements, durch die das Array initialisiert und jedem Element eine Null zugeordnet wird. Wir nennen dieses Array das "A Array" und verwenden A(X) für die indizierte Variable.

```
100 REM*** SCHOKOLADENRIEGEL-ZAEHLER
110 REM***INITIALISIERUNG
120 DIM ..........................................
```

– –

```
120 DIM A(8) : FOR X=1 TO 8 : LET A(X)=0 : NEXT X
```

39. Jetzt wollen wir die Daten paarweise einlesen. Verwenden Sie K für die einzelnen Mitglieder und Q für die Anzahl der Schokoladenriegel und prüfen Sie jeweils, ob Sie bereits am Daten-Ende-Flag angelangt sind. Sofern alle Daten verbraucht sind, soll das Programm zur

Zeile 310 weitergehen. Alle erwähnten Schritte können in einem einzeiligen Mehrfach-Statement untergebracht werden.

```
200 REM***LIES DATEN UND ZAEHLE
210 ..............................................
```

```
210 READ K,Q : IF K=-1 THEN 310
```

40. Jetzt wird es schwieriger. Wir müssen nämlich die Anzahl der Schokoladenriegel jeweils in dem Array-Element aufaddieren, das dem Clubmitglied zugeordnet ist, das sie auch verkauft hat. Stellen Sie sich dazu den Index K in der indizierten Variablen A(K) als die Indentifikationsnummer der einzelnen Mitglieder vor.

```
220 LET A(K)=A(K)+Q : GOTO 210
```

Diese Menge wird zu dem Element hinzuaddiert, das dem Clubmitglied mit der Nummer K zugeordnet ist.

Diese Anweisung bedeutet "gehe zurück zur Zeile 210 und lies weitere Daten!"

Wenn K=2 (Clubmitglied 2) und die Menge Q=4 ist, bewirkt Zeile 220, daß das Array-Element A(...........) um .... größer wird.

A(2); 4

41. Wenn die Array-Elemente, vor dem Lesen und Addieren der zusätzlichen, nachfolgend angegebenen Daten so aussehen, wie links angegeben, wie werden sie anschließend aussehen?

```
920 DATA 4,6,3,8,6,6,7,2,4,3
```

	vorher		nachher
A(1)=	8	A(1)=	
(2)=	4	(2)=	
(3)=	6	(3)=	
(4)=	4	(4)=	

	Vorher		Nachher
(5)=	3	(5)=	
(6)=	2	(6)=	
(7)=	12	(7)=	
(8)=	7	(8)=	

A(1)=	8
(2)=	4
(3)=	14
(4)=	13
(5)=	3
(6)=	8
(7)=	14
(8)=	7

42. Hier sehen Sie das gesamte Programm, soweit wir es bisher erarbeitet haben:

```
100 REM***SCHOKOLADENRIEGELZAEHLER
110 REM***INTITIALISIERE
120 DIM A(8)
130 FOR X=1 TO 8 : A(X)=0 : NEXT X
200 REM***LIES DATEN UND ZAEHLE
210 READ K,Q : IF K=–1 THEN 310
220 LET A(K)=A(K)+Q : GOTO 210
900 REM***DATENPAARE: IDENTIFIKATIONS-NUMMER UND
901 REM***MENGE
910 DATA 4,6,7,12,3,6,1,8,4,5,5,3,8,20
920 DATA 2,4,3,8,6,6,5,10,7,12,8,4,1,3
929 DATA 1,6
930 DATA –1,–1
```

Das Programm startet und addiert auf. Bevor es aber anhält, müssen wir noch vorsehen, daß es die Ergebnisse in einer Aufstellung ausdruckt. Für diese Liste wollen wir folgende Überschrift vorsehen:

```
300 REM***DRUCKE UEBERSCHRIFT
310 PRINT "NUMMER ", "MENGE"
```

Die Ergebnisse sollen mit Hilfe eine FOR-NEXT-Schleife ausgedruckt werden.

```
320 FOR X=1 TO 8
330 PRINT X, A(X)
340 NEXT X
```

oder

```
320 FOR X=1 TO 8 : PRINT X, A(X) : NEXT X
```

Geben Sie, unter Verwendung der Daten im DATA-Statement an, wie die ausgedruckte Liste aussehen wird, nachdem das Programm gelaufen ist.

RUN

....

....

....

....

....

....

....

....

....

```
NUMMER        MENGE
1             17
2             4
3             14
4             11
5             13
6             6
7             24
8             24
READY
```

43. Bis jetzt gibt unsere Liste nur an, wer wieviel verkauft hat, jedoch keine Namen und weder die Gesamtstückzahl noch den Geweinn. Mit einiger Hilfe kann unser Computer aber diese Angaben liefern.

Fangen wir mit der Summe an. In BASIC hat jedes Array ein Element, das wir bisher noch nicht verwendet haben. Es handelt sich um das Nullelement (0). Erinnern Sie sich daran: Wenn Sie ein Array mit 8 Elementen durch DIM(8) dimensionieren, erhalten Sie in Wirklichkeit 9 Elemente, nämlich 0,1,2,3,4,5,6,7,8.

Da wir keinem Club-Mitglied die Nummer "0" zugweiesen haben, wurde das Element des A-Arrays mit dem Index 0 bisher nicht benutzt. Es kann daher zur Ermittlung der Gesamtanzahl verkaufter Schokoladenriegel verwendet werden, obwohl wir auch eine andere Variable vorsehen könnten. Sehen Sie sich das Statement 410 an, durch das die Gesamtzahl der Schokoladenriegel in A(0) aufaddiert wird.

```
400 REM***BERECHNE DIE GESAMTSTUECKZAHL UND DEN
    GEWINN
410 FOR X=1 TO 8 : A(0)=A(0)+A(X) : NEXT X
```

Wenn das A-Array mit 0 beginnt, müssen wir daran denken, das FOR-Statement in der Initialisierungs-Routine so abzuändern, daß die Zuweisung von Nullen jetzt bei A(0) beginnt. Schreiben Sie die Initialisierungs-Zeile für dieses Programm so um, daß alle Array-Elemente 0 als Anfangswert erhalten.

130 ....

```
130 FOR K=0 TO 8 : A(X)=0 : NEXT X
```

44. Das Statement, das alle von den Clubmitgliedern verkauften Schokoladenriegel aufaddiert, verwendet zum akkumulieren der in A(1) bis A(8) gespeicherten Werte das Element A(0) des A-Arrays.

Wenn das Array so aussieht, wie in Abschnitt 41 (nach dem Lesen der Daten), welcher Wert wird dann nach einem Durchlauf der Schleife in Zeile 410 in A(0) gespeichert? ....
Und nach drei Schleifendurchläufen? ....

- -

8; 26

45. Schreiben Sie jetzt die noch fehlenden Programmzeilen 420 und 430, mit denen die Gesamtstückzahl und der Gewinn (55 Pfg. je Schokoladenriegel) ausgedruckt werden sollen. Hier sehen Sie, was Zeile 420 ausdruckt:

GESAMTSTUECKZAHL: 113
GEWINN: 62.15

420 ....
430 ....

- -

```
420 PRINT "GESAMTSTUECKZAHL:"; A(0)
430 PRINT "GEWINN:";A(0)*0.55
```

46. Hier nun ein vollständiges Listing unseres bisherigen Programms, mit einem ausgedruckten Lauf.

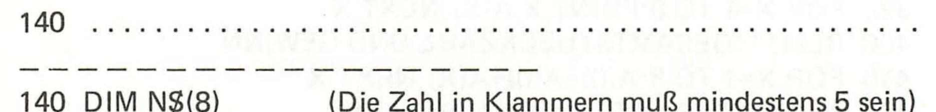
```
100 REM***SCHOKOLADENRIEGELZAEHLER
110 REM***INITIALISIERE
120 DIM A(B)
130 FOR X=0 TO 8:A(X)=0:NEXT X
200 REM***LIES DATEN UND ZAEHLE
210 READ K,Q:IF K=-1 THEN 310
215 IF K=-1 THEN 310
220 A(K)=A(K)+Q:GOTO 210
300 REM***DRUCKE DIE UEBERSCHRIFT
310 PRINT "NUMMER", "MENGE"
320 FOR X=1 TO 8:PRINT X,A(X):NEXT X
400 REM***GESAMTSTUECKZAHL UND GEWINN
410 FOR X=1 TO 8:A(0)=A(0)+A(X):NEXT X
420 PRINT "GESAMTSTUECKZAHL:";A(0)
430 PRINT "GEWINN:";A(0)*0.55
900 REM***DATENPAARE: IDENTIFIKATIONSNUMMER UND
901 REM***MENGE
910 DATA 4,6,7,12,3,6,1,8,4,5,5,3,8,20
920 DATA 2,4,3,8,6,6,5,10,7,12,8,4,1,3
```

```
929 DATA 1,6
930 DATA -1,-1
RUN
NUMMER      MENGE
1           17
2           4
3           14
4           11
5           13
6           6
7           24
8           24
GESAMTSTUECKZAHL: 113
GEWINN: 62.15
```

Das ist schon eine ganz brauchbare Aufstellung. Aber halt! Finden Sie nicht auch, daß Sie ein wenig unpersönlich aussieht? Es wäre noch schöner, wenn die Namen der Mitglieder, statt einer Nummer ausgedruckt würden. Wie können wir das erreichen? Sehr einfach. Wir weisen den Computer an, die Namen aus DATA-Statements zu lesen.

String-Variable müssen, wie wir bereits wissen, genau wie ein Array dimensioniert werden. Vergessen Sie dabei auch nicht das $-Zeichen an der richtigen Stelle, das eine String-Variable identifiziert. Es ist zweckmäßig, jedes Array in der Zeile zu dimensionieren, in der es erstmals verwendet oder initialisiert wird.

Schreiben Sie ein DIM-Statement zur Initialisierung von N$(X), für einen String, der auch für den längsten Namen eines Clubmitglieds ausreicht.


140 ....

----

140 DIM N$(8) (Die Zahl in Klammern muß mindestens 5 sein)

47.Nehmen wir einmal an, wir haben die Namen der Clubmitglieder in der gleichen Reihenfolge in ein DATA-Statement eingesetzt, wie ihre Identifikationsnummern.

```
DATA JERRY, BOBBY, MARY, DANNY, KARL, MIMI, DOUG,
     SCOTT
```

Beachten Sie bitte, daß diese Zeile auf dem Bildschirm in die nächste Zeile hineinreicht. Der Computer stört sich aber nicht daran und sieht Zahlen oder Namen, die am Ende der Zeile getrennt sind, als zusammengehörig an.

Da unser BASIC keine String-Arrays, sondern nur numerische Arrays erlaubt, müssen wir den Computer auffordern, jeden Namen zu dem Zeitpunkt zu lesen und auszudrucken, an dem er die Liste ausgibt. Das Statement zum Lesen der Namen aus dem DATA-Statement muß daher innerhalb der FOR-NEXT-Schleife liegen, die die Liste ausdruckt (Zeilen 320 und 340).

```
300 REM***DRUCKE UND LISTE
310 PRINT
320 PRINT "NAME","MENGE" : PRINT
330 FOR X=1 TO 8 : READ N$
340 PRINT N$,A(X) : NEXT X
```

Beachten Sie, daß hier kein Daten-Ende-Flag vorgesehen wurde, da die FOR-NEXT-Schleife nur achtmal durchlaufen wird und im DATA-Statement auch nur 8 Namen enthalten sind.

Jetzt bleibt noch die Frage offen, wo wir das DATA-Statement einfügen sollen, das die Namen der Clubmitglieder enthält. Denken Sie daran, daß wir eine Fehlermeldung vermeiden möchten, die wir erhalten, wenn ein READ-Statement mit einer numerischen Variablen auf ein DATA-Statement trifft, das Strings enthält.

Wo muß das DATA-Statement mit den Namen der Clubmitglieder also angeordnet werden? . . . .

– – – – – – – – – – – – – – – – – – – –

Am Programmende, da alle numerischen Daten vorher eingelesen werden.

48. Jetzt können Sie einmal zeigen, ob Sie alles verstanden haben. "Spielen" Sie bitte Computer und schreiben Sie auf, wie ein RUN des folgenden Programms aussehen würde.

```
100 REM***SCHOKOLADENRIEGELZAEHLER
110 REM***INITIALISIERUNG
120 DIM A(8)
130 FOR Z=0 TO 8:A(X)=0:NEXT X
140 DIM N$(8)
200 REM***LIES DATEN UND ZAEHLE
210 READ K,Q:IF K=-1 THEN 310
215 IF K=-1 THEN 310
220 A(K)=A(K)+Q:GOTO 210
300 REM***DRUCKE DIE LISTE
310 PRINT
320 PRINT "NAME","MENGE"
330 FOR X=1 TO 8: READ N$
340 PRINT N$,A(X):NEXT X
400 REM***GESAMTSTUECKZAHL UND GEWINN
410 FOR X=1 TO 8:A(0)=A(0)+A(X):NEXT X
420 PRINT "GESAMTSTUECKZAHL:";A(0)
430 PRINT "GEWINN:";A(0)*0.55
900 REM***DATENPAARE: IDENTIFIKATIONSNUMMER UND
901 REM***MENGE
910 DATA 4,6,7,12,3,6,1,8,4,5,5,3,8,20
920 DATA 2,4,3,8,6,6,5,10,7,12,8,4,1,3
929 DATA 1,6
930 DATA -1,-1
940 DATA JERRY,BOBBY, MARY,DANNY,KARL,MIMI,DOUG,
    SCOTT
RUN
.............................................
.............................................
.............................................
.............................................
.............................................
.............................................
.............................................
.............................................
.............................................
.............................................
.............................................
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
RUN
```

```
NAME          MENGE
JERRY         17
BOBBY         4
MARY          14
DANNY         11
KARL          13
MIMI          6
DOUG          24
SCOTT         24
GESAMTSTUECKZAHL: 113
GEWINN: 62.15
```

49. Jetzt wollen wir uns einer anderen Anwendungsmöglichkeit für einfach indizierte Variable zuwenden, bei der weder Stimmen gezählt werden, noch irgendetwas aufaddiert wird.

Sie haben doch sicher schon einmal von der "Partnervermittlung per Computer" gehört und sich bestimmt gefragt, wie das wohl funktioniert? Sie müssen dazu eine Reihe von Fragen beantworten, die dann im Computer gespeichert werden. Andere Personen, die ebenfalls einen Partner suchen, machen dasselbe. Dann werden die Antworten verglichen, um die "Kompatibilität" oder Übereinstimmung zu prüfen. Wir wollen jetzt einmal ein derartiges, allerdings stark vereinfachtes Programm ausarbeiten, das Sie für Ihre eigenen Zwecke abändern können.

Wir verwenden einen Multiple-Choice-Fragebogen mit insgesamt fünf Fragen und jeweils vier möglichen Antworten. (Sie können dabei irgendwelche Fragen stellen, die Ihnen interessant erscheinen.) Die Antworten werden in DATA-Statements gespeichert, jeweils beginnend mit dem Namen, gefolgt von den fünf Antworten. Meine eigenen Antworten sehen Sie in Zeile 910:

```
900 REM***ANTWORTEN
901 REM***NAME GEFOLGT VON DEN ANTWORTEN
902 REM***'SEINE' ANTWORTEN IN ZEILE 910
910 DATA LEROY'3,3,4,2,1
```

Schreiben Sie jetzt DATA-Statements (Zeilen 920 — 970) für die folgenden Personen und ihre Antworten. Sehen Sie außerdem ein Daten-Ende-Flag vor.

	Q1	Q2	Q3	Q4	Q5
JOAN	1	4	2	2	1
TONI	2	2	2	3	3
LAURA	2	3	3	1	2
MARY	3	3	4	2	1
IRENE	3	1	4	2	1

```
920 ..................................................
930 ..................................................
940 ..................................................
950 ..................................................
960 ..................................................
970 DATA END
```

```
920 DATA JOAN,1,4,2,2,1
930 DATA TONI,2,2,2,3,3
940 DATA LAURA,2,3,3,1,2
950 DATA MARY,3,3,4,2,1
960 DATA IRENE,3,1,4,2,1
```

50. Jetzt sollen Sie den Rest des Programms schreiben, wobei wir Ihnen natürlich etwas Hilfestellung geben werden. Zuerst brauchen Sie ein DIM-Statement, das es dem Programm ermöglicht, "seine" im Array C gespeicherten Daten mit "ihren" im Array A abgelegten Daten zu vergleichen. Initialisieren Sie auch eine String-Variable N$ für "seinen" und H$ für "ihren" Namen.

```
100 REM***SUMULATION EINER PARTNERVERMITTLUNG
105 REM***INITIALISIERUNG
110 ..................................................
```

```
110 DIM C(5),A(5),N$(10),H$(10)
```

Beachten Sie bitte: In einem DIM-Statement können Sie mehr als ein Array dimensionieren!

51. Lesen Sie jetzt als nächstes das erste DATA-Statement, das "seinen" Namen enthält und die zugehörigen 5 Antworten ein. Die Antworten

sollten in das C-Array gespeichert werden, und zwar unter Verwendung der in den Abschnitten 8 und 18 angegebenen Methode.

Drucken Sie dann "seinen" Namen und alle Antworten in einer Zeile aus, sodaß Sie sie später mit "ihren" Antworten vergleichen können.

```
200 REM***'SEIN' NAME UND SEINE ANTWORTEN
210 ............................................
220 ............................................
230 ............................................
240 ............................................
```

```
210 READ N$
220 FOR X=1 TO 5:READ C:C(X)=C:NEXT X
230 PRINT N$,
240 FOR X=1 TO 5:PRINT C(X);" "; :NEXT X:PRINT
```

Haben Sie an dieses PRINT-Statement gedacht? Es positioniert den Cursor wieder am Anfang der nächsten Zeile.

Anmerkung:
Möglicherweise haben Sie mehr oder weniger Zeilennummern verwendet. Auf jeden Fall aber müssen die Statements in der angegebenen Weise ausgeführt werden, und zwar in der gleichen Reihenfolge.

52. Lesen Sie als nächstes einen von "ihren" Namen und Datensätzen in das Array A ein. Veranlassen Sie, daß "ihre" Antworten so ausgedruckt werden, daß sie rein visuell mit "seinen" Antworten verglichen werden können. Nach dem Lesen "ihres" Namens in Zeile 310 müssen Sie prüfen, ob bereits das Daten-Ende-Flag (das Wort END) vorliegt. Verwenden Sie dazu den String-Vergleich IF H$="END" THEN END. Die Ausgabe, die wir zu diesem Zeitpunkt wünschen, sollte folgendermaßen aussehen:

```
RUN
LEROY          3    3    4    2    1
JOAN           1    4    2    2    1
```

Vervollständigen Sie bitte das Programm-Segment, damit diese Ausgabe erzeugt wird.

```
300 REM***'IHRE' NAMEN UND ANTWORTEN
310 ...........................................
320 ...........................................
330 ...........................................
340 ...........................................
```

```
310 READ H$:IF H$="END" THEN END
320 FOR X=1 TO 5:READ A:A(X)=A:NEXT X
330 PRINT H$,
340 FOR X=1 TO 5:PRINT A(X);" ";:NEXT X:PRINT
```

53. Nun kommt der schwierigste Teil, in dem der Inhalt des Arrays C mit dem Inhalt des Arrays A verglichen und die Anzahl der Übereinstimmungen in die Variable M addiert wird. Denken Sie darüber nach, wie diese Schritte zu programmieren sind und vervollständigen Sie das Programm so, daß ein Lauf des vollständigen Programms aussieht, wie hier abgebildet:

```
RUN
LEROY          3       3       4       2       1
JOAN           1       4       2       2       1
  2 UEBEREINSTIMMUNGEN
TONI           2       2       2       3       3
  0 UEBEREINSTIMMUNGEN
LAURA          2       3       3       1       2
  1 UEBEREINSTIMMUNGEN
MARY           3       3       4       2       1
  5 UEBEREINSTIMMUNGEN
IRENE          3       1       4       2       1
  4 UEBEREINSTIMMUNGEN
```

```
400 REM***VERGLEICHE UND ZAEHLE DIE
401 REM***'UEBEREINSTIMMUNGEN'
410 M=0
```

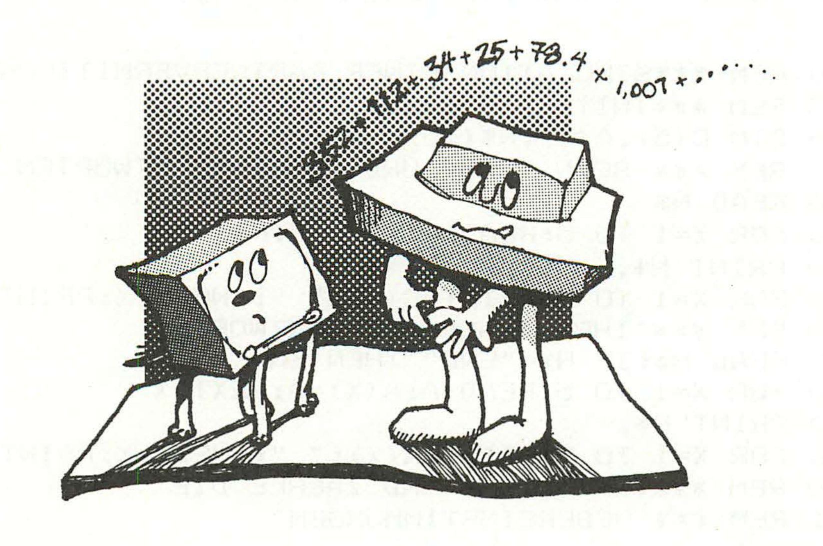
```
420 FOR X=1 TO 5
430 IF ......... THEN 450 ◄──────────── Vergleiche
440 LET M= ............ ◄──── Addiere die Übereinstimmungen
450 .......... ◄──── Schließen der Schleife, um einen wei-
                     teren Vergleich durchzuführen.
460 PRINT ........... "UEBEREINSTIMMUNGEN"
470 PRINT
480 GOTO ............. ◄── Einlesen eines weiteren Datensatzes
```

```
410 M=0
420 FOR X=1 TO 5
430 IF C(X) <> A(X) THEN 450
440 M=M+1
450 NEXT X
460 PRINT M;" UEBEREINSTIMMUNGEN"
470 PRINT
480 GOTO 310
```

Nachfolgend finden Sie ein Listing des gesamten Programms:

```
100 REM ***SIMULATION EINER PARTNERVERMITTLUNG
103 REM ***INITIALISIERUNG
110 DIM C(5),A(5),N$(10),H$(10)
200 REM ***'SEIN' NAME UND 'SEINE' ANTWORTEN
210 READ N$
220 FOR X=1 TO 5:READ C:C(X)=C:NEXT X
230 PRINT N$,
240 FOR X=1 TO 5:PRINT C(X);" ";:NEXT X:PRINT
300 REM ***'IHRE' NAMEN UND ANTWORTEN
310 READ H$:IF H$="END" THEN END
320 FOR X=1 TO 5:READ A:A(X)=A:NEXT X
330 PRINT H$,
340 FOR X=1 TO 5:PRINT A(X);" ";:NEXT X:PRINT
400 REM ***VERGLEICHE UND ZAEHLE DIE
401 REM ***'UEBEREINSTIMMUNGEN'
410 M=0
420 FOR X=1 TO 5
430 IF C(X)<>A(X) THEN 450
440 M=M+1
450 NEXT X
```

```
460 PRINT M;" UEBEREINSTIMMUNGEN"
470 PRINT
480 GOTO 310
900 REM ***ANTWORTEN AUF DIE FRAGEN
901 REM ***NAME GEFOLGT VON DEN ANTWORTEN
902 REM ***'SEINE' ANTWORTEN IN ZEILE 910
910 DATA LEROY,3,3,4,2,1
920 DATA JOAN,1,4,2,2,1
930 DATA TONI,2,2,2,3,3
940 DATA LAURA,2,3,3,1,2
950 DATA MARZ,3,3,4,2,1
960 DATA IRENE,3,1,4,2,1
970 DATA END
```



EIGENTEST

Wenn Sie den nachfolgenden Test über indizierte Variable vollständig durcharbeiten, sind Sie ausreichend für das nächste Kapitel vorbereitet, das Ihre Programmierfähigkeiten nochmals erweitern wird. In diesem Kapitel werden Sie nämlich die Verwendung noch komplexerer, indizierter Variabler erlernen. Es ist daher sehr wichtig, daß Sie alle Informationen aus diesem Kapitel parat haben.

1. Geben Sie an, welche der folgenden Variablen zulässige, indizierte BASIC-Variable sind.

   a) X ....   b) X2 .....   c) X(2) ....
   d) 2(X) ...   e) XX(2) ....   f) X(K) .....
   g) X_2 ....   h) (I-J .....

2. Nehmen Sie an, daß den Variablen Werte zugeordnet wurden, wie nachfolgend angegeben. Beachten Sie, daß sowohl einfache, als auch indizierte Variable gezeigt werden.

Q	2	A(1)	37
A	3	A(2)	4
A1	1	A(3)	23
		A(4)	19

   Denken Sie daran, daß A, A1 und A(1) verschiedene Variable sind. Tragen Sie anschließend den Wert jeder Variablen ein.

   a) A(Q) ....
   b) A(A) ....
   c) A(A1) ....
   d) A(A(2)) ....
   e) A(A(Q)) ....
   f) A(A–Q) ....
   g) A(A+1) ....
   h) A(2*Q) ....

3. Was wird beim Lauf des folgenden Programms ausgedruckt?

```
100 REM***GEHEIMNIS-PROGRAMM
110 DIM X(10)
```

```
120 READ N
130 FOR K=1 TO N
140 READ X:X(K)=X
150 NEXT K
160 FOR K=1 TO N
170 IF X(K) < 0 THEN 190
180 PRINT X(K);" ";
190 NEXT K
900 REM***WERTE FUER DAS N- UND X-ARRAY
910 DATA 7
920 DATA 23,-44,37,0,-12,- 58,87
RUN
```

. . . .

4. Welches ist im Programm aus Frage 3 der größte Wert von N, für den das Programm benutzt werden kann? . . . .

   Was würde geschehen, wenn wir versuchten, das Programm unter Verwendung der folgenden Daten laufen zu lassen? . . . .

   . . . .

```
910 DATA 12
920 DATA 3,6,-2,0,9,0,7,3,-5,4,-1,7
```

5. Modifizieren Sie das Stimmen-Zählprogramm aus Abschnitt 27 so, daß die gesamte Stimmenzahl beider Kandidaten ebenfalls ausgedruckt wird. Ein Ausdruck könnte dann zum Beispiel so aussehen:

```
RUN
HANS SCHMITT: 19
CLAUDIA MUELLER: 16

GESAMTSTIMMEN: 35
```

   . . . .

   . . . .

6. Modifizieren Sie das Stimmenzählprogramm aus Abschnitt 27 so, daß der Stimmanteil in Prozent von der Gesamtzahl angegeben wird, und zwar abgerundet auf das nächste ganze Prozent.

```
RUN
HANS SCHMITT: 54%
CLAUDIA MUELLER: 46%
```

   . . . .

. . . .

. . . .

. . . .

. . . .

7. Sie wurden damit beauftragt, eine Wohltätigkeits-Sammlung in Ihrer Nachbarschaft durchzuführen. Sie haben 5 Helfer, die von Tür zu Tür gehen, Geld sammeln und es am Ende jedes Tages bei Ihnen abliefern. Sie tragen ihre Namen und die gesammelten Beträge in ein Formular ein. Diese Daten sollen dann zu weiteren Bearbeitung in Ihren Computer eigegeben werden. Schreiben Sie daher ein BASIC-Programm, das den von jedem Helfer jeweils abgelieferten Betrag aufaddiert, außerdem aber auch die von allen insgesamt gesammelte Summe ausdruckt. Ihre Aufstellung sollte die Namen jedes Helfers und die gesammelten Beträge enthalten.

```
RUN
NAME        GESAMMELTER BETRAG
FRED        125
JOANN       205
MARYJO      100
JERRY       100
BOB         200
SUMME       730
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

8. Die Mitglieder Ihres Computer-Clubs möchten durch einen Test entscheiden, wer der beste Programmierer ist. Sie werden damit beauftragt, einen Multiple-Choice-Test über Programmier-Konzepte durchzuführen und den Test unter Verwendung des Computers zu korrigieren. Die möglichen Antworten sind die Zahlen 1,2,3,4 oder 5. Der Test selbst enthält 10 Fragen. Sie geben zunächst die 10 korrekten Antworten in die ersten DATA-Statements des Programms ein. Sie werden anschließend in das Array K eingelesen. In den nachfolgenden DATA-Statements geben Sie erst den Namen des Club-Mitglieds ein, gefolgt von den 10 Antworten, die bei dem Test gegeben wurden. (Die Eingabe erfolgt in das Array R). Ihre Aufgabe besteht darin, ein BASIC-Programm zu schreiben, das die Tests korrigiert und eine Aufstellung ausdruckt, die ähnlich wie das nachfolgende Beispiel aussieht.

```
RUN
NAME      TREFFER
DANNY     5
KARL      5
MIRIAM    4
SCOTT     7
```

. . . .

. . . .

9. Haben Sie sich schon jemals gefragt, welche Chancen Sie bei einem Spiel haben, bei dem Würfel verwendet werden? Wenn Sie ein Programm schreiben, das einen rollenden Würfel sumuliert und dabei zählen, wie oft die einzelnen Zahlen von 1 bis 6 "gewürfel" werden, bekommen Sie eine ziemlich präzise Vorstellung von Ihren Gewinnmöglichkeiten.

Schreiben Sie daher ein BASIC-Programm, durch das 1000 mal das Werfen eines Würfels simuliert wird. Addieren Sie nach jedem Wurf das Auftreten der jeweiligen Zahl in einem zugehörigen Array-Element. Drucken Sie nach 1000 Würfen die Ergebnisse in Form einer Liste aus, die angibt, wie oft jede Zahl während der Computer-Simulation aufgetreten ist. Ihre Liste sollte so aussehen, wie nachfolgend abgedruckt.

```
RUN
WIEVIELE WUERFE? 1000
ZAHL        WIE OFT AUFGETRETEN?
1           159
2           152
3           173
4           142
5           189
6           185
```

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

. . . .

Antworten zum Eigentest

Die Zahlen in den Klammern geben die Abschnitte an, in denen der jeweilige Stoff behandelt wurde.

1. c), f) und h) sind gültige, indizierte Variable.
   Längere Variablen-Namen, wie zum Beispiel in e) können eventuell bei Ihrem Computer zulässig sein. Sehen Sie dazu in Ihrem Handbuch nach. (Abschnitt 1)

2. a) 4
   b) 23
   c) 37
   d) 19 A(A(2)) = A(4) = 19
   e) 19 A(A(Q)) = A(A(2)) =A(4) = 19
   f) 37 A(3–2) = A(1) = 37
   g) 19 A(3+1) = A(4) = 19
   h) 19 A(2+2) = A(4) = 19 (Abschnitt 4)

3. 23 37 0 87 (Abschnitte 7 und 8)

4. 10. Der Computer würde eine Fehlermeldung ausdrucken.
   (Abschnitte 11 und 12)

5. Fügen Sie die folgenden Statements hinzu:
```
330 PRINT
340 PRINT "GESAMTSTIMMEN:";C(1)+C(2)
```
   (Abschnitte 43 – 46)

6. Führen Sie, beginnend mit Zeile 310, folgende Änderungen durch:
```
310 LET T=C(1)+C(2)
320 LET S=INT(100*C(1)/T + .5)
330 LET G=INT(100*C(2)/T + .5)
340 PRINT "HANS SCHMITT:";S;"%"
350 PRINT "CLAUDIA MUELLER:";G;"%"
```
   (Abschnitt 27 und Kapitel 4)

7.
```
10 REM
15 REM
20 DIM N$(10), T(5):G=0:FOR X=1 TO 5:T(X)=0:NEXT X
25 REM
30 READ P,D:IF P=–1 THEN 60
```

```
40 T(P)=T(P)+D:G=G+D:GOTO 30
50 REM
60 PRINT "NAME","GESAMMELTER BETRAG"
70 FOR P=1 TO 5:READ N$:PRINT N$,T(P):NEXT P
80 PRINT "SUMME:",G
900 REM
910 DATA 2,45,1,75,3,25,4,100,3,25
920 DATA 5,125,3,50,1,50,2,120,2,40
930 DATA 5,75,-1,-1
940 DATA FRED, JOANN,MARYJO,JERRY,BOB
```

(Abschnitte 35 – 53)

8.

```
10 REM
20 REM
30 DIM K(10),R(10),N$(10)
40 REM
50 PRINT "NAME","TREFFER":PRINT
60 REM
70 FOR X=1 TO 10
80 READ K:K(X)=K
90 NEXT X
100 REM
110 READ N$:IF N$="END" THEN END
120 FOR X=1 TO 10
130 READ R:R(X)=R:NEXT X
140 S=0
150 REM
160 FOR X=1 TO 10
170 IF K(X)<>R(X) THEN 190
180 S=S+1
190 NEXT X
200 PRINT N$,S:GOTO 110
900 REM
910 DATA 3,4,3,3,5,1,2,3,2,1
920 REM
930 DATA DANNY,1,2,3,4,5,1,2,3,4,5
940 DATA KARL,1,3,2,4,1,2,3,2,1,5
950 DATA MIRIAM,3,2,2,1,4,1,2,3,1,2
960 DATA SCOTT,1,2,3,3,5,1,2,3,2,2
970 DATA END
```

(Abschnitte 49 – 53)

9.
```
10 REM
20 DIM D(6)
30 FOR X=1 TO 6:D(X)=0:NEXT X
40 PRINT "WIEVIELE WUERFE?";
50 INPUT R
60 FOR X=1 TO R
70 N=INT(6*RND(1))+1:D(N)=D(N)+1
80 NEXT X
90 PRINT "ZAHL","WIE OFT AUFGETRETEN?"
100 REM
110 FOR X=1 TO 6
120 PRINT X,D(X)
130 NEXT X
```

(Abschnitte 25 – 31)

KAPITEL 8

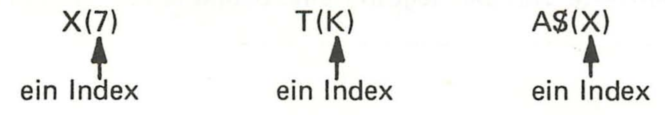
Doppelt-indizierte Variable

In Kapitel 7 wurden Sie mit einfach-indizierten Variablen und ihren zahlreichen Anwendungsmöglichkeiten vertraut gemacht. In diesem Kapitel werden Sie auf keine neuen Statements treffen, sondern einige neue Anwendungen und Variationen von Dingen lernen, die Sie bereits kennen.

In diesem Kapitel wollen wir Ihre Kenntnisse in ATARI-BASIC auf Variable mit zwei Indizes erweitern, die wir als "doppelt-indizierte" Variable bezeichnen. Doppelt-indizierte Variable werden für numerische Arrays verwendet, die mehrere Spalten und Reihen erfordern würden, wie beispielsweise komplexen Wahlanalysen, detaillierte Finanz-Analyse-Probleme und eine Vielzahl von Anwendungen bei Brettspielen. Wenn Sie dieses Kapitel beendet haben, können Sie . . . .

- Variable mit zwei Indizes benutzen;
- doppelt-indizierten Variablen in einem zweidimensionalen Array (auch als Matrix bezeichnet) Werte zuweisen;
- das DIM-Statement verwenden, um dem Computer die Dimensionen eines zweidimensionalen Arrays mitzuteilen.

1. In Kapitel 7 verwendeten wir indizierte Variable, wie z.B. X(7) und T(K). Dabei handelt es sich um einfach-indizierte Variable. Das heißt, jede Variable hat einen einzigen Index.



In diesem Kapitel werden wir doppelt-indizierte Variable verwenden, d.h. Variable, die zwei Indizes aufweisen.



T(3) ist eine indizierte Variable mit wievielen Indizes? ....
T(7,5) ist eine indizierte Variable mit wievielen Indizes? ....

1; 2

2. Es ist üblich, sich doppelt-indizierte Variable in einem Feld von Spalten und Reihen angeordnet zu denken, wie es nachfolgend gezeigt ist.

	Spalte 1	Spalte 2	Spalte 3	Spalte 4
Reihe 1	A(1,1)	A(1,2)	A(1,3)	A(1,4)
Reihe 2	A(2,1)	A(2,2)	A(2,3)	A(2,4)
Reihe 3	A(3,1)	A(3,2)	A(3,3)	A(3,4)

Das obige Array hat .... Reihen und .... Spalten.

3; 4

3. Durch die Anordnung in Abschnitt 2 können wir Indizes zu speziellen Positionen innerhalb des Arrays ("Schachteln"), also in den jeweiligen Reihen und Spalten, in Beziehung setzen. Sie werden "Elemente" des Arrays genannt. Beispielsweise:

A(2,3)

Reihe ——— Spalte

A(1,1) liegt in Reihe 1, Spalte 1; A(1,2) liegt in Reihe 1, Spalte 2. Welche indizierte Variable liegt in Reihe 3, Spalte 2? ....

A(3,2)

4. Die Anordnung der doppelt-indizierten Variablen in Abschnitt 2 wird als Tabelle, "Matrix" oder auch zweidimensionales Array bezeichnet.

In Kapitel 7 haben wir Arrays von einfach-indizierten Variablen vorgestellt, die auch als Liste, "Vektoren" oder eindimensionale Arrays bezeichnet werden.

Dies ist eine Liste:	X(1)	X(2)	X(3)
Dies ist eine Tabelle:	C(1,1)	C(1,2)	C(1,3)
	C(2,1)	C(2,2)	C(2,3)
	C(3,1)	C(3,2)	C(3,3)

a) Eine Liste wird auch als .... oder als .... bezeichnet.
b) Eine Tabelle wird auch als .... oder als .... bezeichnet.

---

a) Vektor, eindimensionales Array
b) Matrix, zweidimensionales Array

5. Eine doppelt-indizierte Variable ist einfach der Name eines Speicherplatzes innerhalb des Computers. Wie bei jeder anderen Variablen, können Sie sich darunter den Namen für eine "Schachtel" zum Abspeichern einer Zahl vorstellen. Hier sehen Sie eine Tabelle mit doppelt-indizierten Variablen.

B(1,1)		B(1,2)		B(1,3)	
B(2,1)		B(2,2)		B(2,3)	

Tun Sie einmal so, als wären Sie der Computer und ordnen Sie der Variablen B(2,1) den Wert 73 zu. Mit anderen Worten, nehmen Sie Ihren Bleistift und schreiben Sie die Zahl 73 in die Schachtel mit der Bezeichnung B(2,1). Machen Sie dann das gleiche für die folgenden Variablen.

```
LET B(1,3) = 0
LET B(1,1) = 49
LET B(2,3) = B(2,1) – B(1,1)
LET B(1,2) = 2*B(2,1)
LET B(2,2) = INT(B(2,1)/B(2,3)
```

B(1,1)	49	B(1,2)	146	B(1,3)	0
B(2,1)	73	B(2,2)	3	B(2,3)	24

6. Wie wir in Kapitel 7 gelernt haben, können Indizes Variable sein. Die indizierte Variable P(R,C) hat variable Indizes.

Wenn R = 1 und C = 2 ist, dann ist P(R,C) = P(1,2).
Wenn R = 4 und C = 3 ist, dann ist P(R,C) = P(4,3).
Wenn R = 7 und C = 3 ist, dann ist P(R,C) = ...

P(7,5)

7. Wir wollen einmal annehmen, daß die folgenden Werte (in den "Schachteln") den entsprechenden Variablen zugewiesen wurden. Beachten Sie, daß sowohl einfache, als auch indizierte Variable vorkommen.

R	2	T(1,1)	7	T(1,2)	0	T(1,3)	–12
C	3	T(2,1)	9	T(2,2)	5	T(2,3)	8
A	1	T(3,1)	16	T(3,2)	13	T(3,3)	10
B	2						

Tragen Sie für jede der folgenden Variablen jeweils den Wert und den Index ein.

T(2,3)= ....
R= ....
C= ....
T(R,C)= ....
T(A,B)= ....

T(1,1)= ....
A= ....
T(A,A)= ....
T(B,R)= ....
T(R,A)= ....
T(R=1,C–2)= ....

---

8 7
2 1
3 7
8 5
0 9
16 T(R+1,C–2)=T(2+1,3–2)=T(3,1)

8. Jetzt wird wieder einmal gewählt! Vielleicht sollten sie, bevor Sie sich damit beschäftigen, noch einmal die entsprechenden Abschnitte im Kapitel 7 nachlesen, die sich mit der Wählerbefragung befassen.

Der nachfolgende "Fragebogen" erfordert zwei Antworten:

Frage 1: Für wen werden Sie in der kommenden Wahl Ihre Stimme abgeben? Kreisen Sie die entsprechende Zahl links von Ihrem Kandidaten ein. 1. Hans Schmitt 2. Claudia Müller 3. Keine Meinung Frage 2: In welcher Altersgruppe sind Sie? Kreisen Sie die Zahl links von Ihrer Altersgruppe ein. 1. Unter 30 2. 30 oder darüber

Da zwei Fragen gestellt wurden, besteht jede Antwort aus zwei Zahlen: der Antwort auf Frage 1 und der Antwort auf Frage 2. Zur Kennzeichnung der beiden Zahlen wollen wir in Zukunft S (für Stimme) zur Darstellung der beiden Antworten auf die Frage 1 und A (für Alter) zur Darstellung der Antwort auf Frage 2 verwenden.

S,A

Antwort auf Frage 1 → S A ← Antwort auf Frage 2

Die möglichen Werte für S sind 1,2 oder 3. Welches sind die möglichen Werte für A? ....

----

1 oder 2

9. Wir haben einige Fragebogen verschickt und folgende typische Antworten erhalten:

Antwort	Bedeutung
1,1	eine Stimme für Hans Schmitt, Wähler unter 30

1,2	eine Stimme für Hans schmitt, Wähler ist 30 Jahre oder älter
3,1	keine Meinung, Wähler ist jünger als 30

Was bedeutet 2,2? ....

Eine Stimme für Claudia Müller, der Wähler ist 30 Jahre oder älter.

10. Wir wollen jetzt ein Programm schreiben, um die Daten eines Fragebogens mit zwei Fragen zu verarbeiten. Wir verwenden indizierte Variable, um die Stimmen zu zählen, so wie es nachfolgend gezeigt ist.

	unter 30		30 oder älter	
Hans Schmitt	C(1,1)	☐	C(1,2)	☐
Claudia Müller	C(2,1)	☐	C(2,2)	☐
Keine Meinung	C(3,1)	☐	C(3,2)	☐

C(1,1) nimmt also die Stimmen für Hans Schmitt auf, die von befragten Personen unter 30 Jahren abgegeben wurde. C(1,2) enthält die Summe aller Stimmen für Hans Schmitt, die er von Wählern über 30 Jahren erhielt. C(2,1) nimmt die Gesamtzahl der Stimmen für .... auf, die ihr von Personen .... gegeben wurden.

Welche indizierte Variable enthält die Anzahl der Stimmen von Wählern der Altersgruppe "30 Jahre oder älter", die keine Meinung hatten? ....

Claudia Müller; unter 30; C(3,2)

11. Hier sind 20 Antworten für unseren Fragebogen. Denken Sie daran, daß jede Antwort ein Paar von Zahlen ist und eine Stimme repräsentiert. Die erste Zahl jedes Paares ist die Antwort auf Frage 1. Die zweite Zahl jedes Paares ist die Antwort auf Frage 2.

3,1	2,2	3,2	1,2	1,2
2,1	2,2	1,1	1,2	3,1
3,2	2,2	3,1	2,1	2,2
1,1	1,1	1,2	1,1	2,1
2,1	1,2	2,1	3,1	2,1

3,1 2,1 3,1 2,2

Schreiben Sie in jedes Feld die entsprechenden Stimmen:

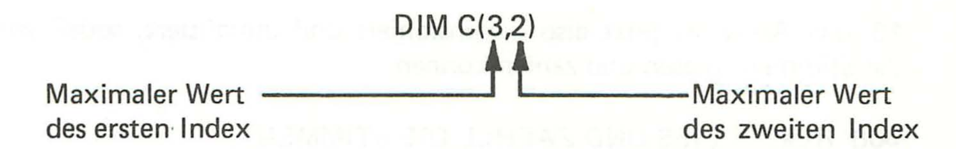
	unter 30		30 oder darüber	
Hans Schmitt	C(1,1)		C(1,2)	
Claudia Müller	C(2,1)		C(2,2)	
Keine Meinung	C(3,1)		C(3,2)	

C(1,1)	4	C(2,2)	5
C(2,1)	7	C(2,2)	5
C(3,1)	6	C(3,2)	2

12. Natürlich möchten wir gern, daß der Computer die Aufgabe des Zählens übernimmt. Nachfolgend sehen Sie den Anfang unseres Programms.

```
100 REM***STIMMENZAEHLUNG, 2 FRAGEN
110 DIM C(3,2)
```

Das DIM-Statement in Zeile 110 definiert ein Array mit maximal 3 Reihen und 2 Spalten. Das heißt, das DIM-Statement dimensioniert ein Array von doppelt-indizierten Variablen, in dem der größte Wert des ersten Index 3 und der größte Wert des zweiten Index 2 ist. Sie müssen doppelt-indizierte Arrays immer dimensionieren, sonst erhalten Sie eine Fehlermeldung.



Als nächstes müssen wir alle Zählvariablen auf Null setzen. Das heißt, wir weisen C(1,1), C(1,2) usw. bis C(3,2) 0 zu. Obwohl dies manche BASIC-Versionen automatisch tun, entspricht es guter Programmier-Praxis, jedes Programm zu initialisieren. Vervollständigen Sie bitte diesen Teil des Programms.

```
200 REM***INITIALISIERUNG: SETZE ALLE ZAEHLER AUF
201 REM***NULL
.......................................................
.......................................................
.......................................................
.......................................................
.......................................................
```

Hier sind vier Lösungsmöglichkeiten:

Methode 1

```
210 LET C(1,1)=0
220 LET C(1,2)=0
230 LET C(2,1)=0
240 LET C(2,2)=0
250 LET C(3,1)=0
260 LET C(3,2)=0
```

Methode 2

```
210 FOR K=1 TO 3
220 LET C(K,1)=0
230 LET C(K,2)=0
240 NEXT K
```

Methode 3

```
210 FOR K=1 TO 3
220 FOR L=1 TO 2
230 LET C(K,L)= 0
240 NEXT L
250 NEXT K
```

Methode 4

```
210 FOR K=1 TO 3:FOR L=1 TO 2:LET C(',L)=0:NEXT L:NEXT K
```

Wir werden immer die Methoden 3 und 4 verwenden, da sie sich leicht für Arrays mit unterschiedlichen Größen verallgemeinern lassen. Bei Verwendung von Methode 3 können wir Zeilen hinzufügen, indem wir Zeile 210 Ändern. Die Anzahl der Spalten kann durch Änderung von Zeile 220 erreicht werden. Natürlich müßten wir gleichzeitig auch das DIM-Statement korrigieren.

13. Das Array ist jetzt also dimensioniert und initialisiert, sodaß wir die Stimmen enlesen und zählen können.

```
300 REM***LIES UND ZAEHLE DIE STIMMEN
310 READ S,A : IF S=-1 THEN 410
320 LET C(S,A)=C(S,A)+1 : GOTO 310
```

Zeile 320 enthält das Zähl-Statement. Es addiert jeweils eine Stimme zu dem durch S und A spezifizierten Array-Element. Nehmen Sie an, daß das Array, vor Ausführung der Zeilen 310 und 320, wie folgt aussieht.

C(1,1)	0	C(2,2)	4
C(2,1)	2	C(2,2)	7
C(3,1)	5	C(3,2)	0

Tragen Sie die folgende Abbildung ein, wie das Array nach der Verarbeitung dieser zusätzlichen Daten aussehen würde:

910 DATA 3,1,2,2,3,2,1,2

C(1,1)		C(1,2)	
C(2,1)		C(2,2)	
C(3,1)		C(3,2)	

C(1,1)	0	C(1,2)	5
C(2,1)	2	C(2,2)	8
C(3,1)	6	C(3,2)	1

14. Da Zeile 310 ein READ-Statement ist, müssen auch noch einige DATA-Statements vorgesehen werden. Hier sind Sie! Sie enthalten die in Abschnitt 1 angegebenen Daten.

```
900 REM***DATENPAARE:
901 REM***STIMMEN GEFOLGT VON DER ALTERSGRUPPE
910 DATA 3,1, 2,2, 3,2, 1,2, 1,2, 2,1
920 DATA 2,2, 1,1, 1,2, 3,1, 3,2, 2,2
930 DATA 3,1, 2,1, 2,2, 1,1, 1,1, 1,2
940 DATA 1,1, 2,1, 2,1, 1,2, 2,1, 3,1
950 DATA 2,1, 3,1, 2,1, 3,1, 2,2, –1,–1
```

Bitte denken Sie daran, daß jede Antwort ein Zahlenpaar ist, aber nur eine Stimme repräsentiert. Um dies zu verdeutlichen, haben wir nach jedem Wertepaar einen Zwischenraum in den obigen DATA-Statements vorgesehen. Warum wird das Flag –1, –1 statt nur –1 verwendet? . . . .

. . . .

Findet der Computer nicht gleichzeitig für die Variablen S und A in Zeile 310 einen Wert vor, druckt er eine Fehlermeldung aus und stoppt.

15. Jetzt bleibt nur noch eine Aufgabe übrig, nämlich die Ergebnisse auszudrucken. Für die in Abschnitt 14 angegebenen Daten sollten die Ergebnisse, beim Lauf des Programms, so ausgegeben werden, wie es das nachfolgende Beispiel zeigt:

```
RUN
KANDITAT        UNTER 30      30 +

SCHMITT         4             5
MUELLER         7             5
KEINE M.        6             2
```

Vervollständigen Sie jetzt das Programm-Segment zum Ausdrucken der Ergebnisse, – C(1,1), C(1,2) usw.

```
400 REM***DRUCKE DIE ERGEBNISSE
..........................................................
..........................................................
..........................................................
..........................................................
..........................................................
```

_ _

Wir haben die Aufgabe folgendermaßen gelöst:

```
410 PRINT "KANDITAT","UNTER 30","30 +"
420 PRINT
430 PRINT "SCHMITT",C(1,1),C(1,2)
440 PRINT "MUELLER",C(2,1),C(2,2)
450 PRINT "KEINE M.",C(3,1),C(3,2)
```

16. Hier ist das Listing des vollständigen Programms:

```
100 REM***WAEHLERBEFRAGUNG, 2 FRAGEN
110 DIM C(3,2)
200 REM***INITIALISIERUNG
210 FOR K=1 TO 3:FOR L=1 TO 2
220 C(K,L)=0
230 NEXT L:NEXT K
300 REM***LIES UND ZAEHLE DIE DATEN
310 READ S,A:IF S=-1 THEN 410
320 C(S,A)=C(S,A)+1:GOTO 310
```

```
400 REM***DRUCKE DIE ERGEBNISSE
410 PRINT "KANDIDAT","UNTER 30","30 +"
420 PRINT
430 PRINT "SCHMITT",C(1,1),C(2,1)
440 PRINT "MUELLER",C(2,1),C(2,2)
450 PRINT "KEINE M.",C(3,1),C(3,2)
900 REM***DATENPAARE:
901 REM***STIMMEN GEFOLGT VON DER ALTERSGRUPPE
910 DATA 3,1,2,2,3,2,1,2,1,2,2,1
920 DATA 2,2,1,1,1,2,3,1,3,2,2,2
930 DATA 3,1,2,1,2,2,1,1,1,1,1,2
940 DATA 1,1,2,1,2,1,1,2,2,1,3,1
950 DATA 2,1,3,1,2,1,3,1,2,2,–1,–1
```

Nehmen wir einmal an, der Fragebogen hätte folgendermaßen ausgesehen:

> Frage 1:
> Wem würden Sie in der kommenden Wahl Ihre Stimme geben? Kreisen Sie die Zahl links von Ihrem Kandidaten ein.
>
> 1. HANS SCHMITT
> 2. CLAUDIA MUELLER
> 3. KEINE MEINUNG
>
> Frage 2:
> Welcher politischen Partei stehen Sie nahe?
>
> 1. USB
> 2. LDS
> 3. SONSTIGE

Modifizieren Sie das Stimmenzähl-Programm so, daß die Antworten wie folgt gezählt werden:

	USB	LDS	SONSTIGE
Hans Schmitt	C(1,1)	C(1,2)	C(1,3)
Claudia Müller	C(2,1)	C(2,2)	C(2,3)
Keine Meinung	C(3,1)	C(3,2)	C(3,3)

Sie müssen dazu die Zeilen 110, 210, 410, 430, 440 und 450 ändern.

110 ....
210 ....
410 ....
430 ....
440 ....
450 ....

— — — — — — — — — — — — — — — — — — — —

```
110 DIM C(3,3)
210 FOR K=1 TO 3:FOR L=1 TO 3
410 PRINT "KANDIDAT","USB","LDS","SONSTIGE"
430 PRINT "SCHMITT",C(1,1),C(1,2),C(1,3)
440 PRINT "MUELLER",C(2,1),C(2,2),C(2,3)
450 PRINT "KEINE M.",C(3,1),C(3,2),C(3,3)
```

Bitte beachten Sie: obwohl wir den Fragebogen geändert haben, ist es nicht erforderlich, das Zähl-Statement in Zeile 320 zu ändern!

17. Hier ist ein Listing des modifizierten Programms. Es enthält auch die neuen DATA-Statements für die Befragung aus Abschnitt 16. Bitte schreiben Sie im Anschluß an das Programm hin, was ausgedruckt wird.

```
100 REM*** STIMMENZAEHLUNG, 2 FRAGEN
110 DIM C(3,2)
200 REM***INITIALISIERUNG
210 FOR K=1 TO 3:FOR L=1 TO 2
220 C(K,L)=0
230 NEXT L:NEXT K
300 REM***LIES UND ZAEHLE DIE DATEN
310 READ S,A:IF S=–1 THEN 410
320 C(S,A)=C(S,A)+1:GOTO 310
400 REM***DRUCKE DIE ERGEBNISSE
410 PRINT "KANDIDAT","USB","LDS","SONSTIGE"
420 PRINT
430 PRINT "SCHMITT",C(1,1),C(1,2),C(1,3)
440 PRINT "MUELLER",C(2,1),C(2,2),C(2,3)
450 PRINT "KEINE M.",C(3,1),C(3,2),C(3,3)
900 REM***DATENPAARE:
```

```
901 REM***STIMMEN GEFOLGT VON PARTEIANGABEN
910 DATA 3,1,1,1,1,2,2,3,1,3,3,1,2,1
920 DATA 2,2,3,2,3,3,1,1,2,3,1,2,2,1
930 DATA 3,3,3,2,3,2,2,2,1,1,1,1,3,2
940 DATA 3,2,2,1,3,1,1,2,1,3,2,2,2,1
950 DATA 3,1,1,3,1,3,2,3,3,3,1,2,1,3
960 DATA 2,3,2,1,3,1,1,2,3,3,2,3,2,1
970 DATA 3,1,3,3,-1,-1
RUN
.......................................................
.......................................................
.......................................................
.......................................................
-----------------------

KANDIDAT      USB      LDS       SONSTIGE

SCHMITT       4        5         5
MUELLER       6        3         5
KEINE M.      6        5         5
```

18. Sehen Sie sich noch einmal diesen Abschnitt des Programms an:

```
430 PRINT "SCHMITT",C(1,1),C(1,2),C(1,3)
440 PRINT "MUELLER",C(2,1),C(2,2),C(2,3)
450 PRINT "KEINE M.",C(3,1),C(3,2),C(3,3)
```

Wir haben hier die indizierten Variablen mit den tatsächlichen Werten für die Indizes hingeschrieben. Statt dessen hätten wir auch FOR-NEXT-Schleifen verwenden können, um die Ergebnisse für die beiden Kandidaten und "Keine Meinung" auszudrucken. Dazu würden wir drei Zeilen mit Mehrfach-Statements benötigen. Denken Sie einmal sorgfältig darüber nach und schreiben Sie die drei Zeilen 430, 440 und 450 so um, daß zum Ausdruck der im C-Array gespeicherten Werte FOR-NEXT-Schleifen benutzt werden. Der Programmlauf sollte genau wie zuvor aussehen.

```
430 ...............................................
440 ...............................................
450 ...............................................
```

```
430 PRINT "SCHMITT",:FOR X=1 TO 3:PRINT C(1,X),:NEXT:
    PRINT
440 PRINT "MUELLER",:FOR X=1 TO 3:PRINT C(2,X),:NEXT:
    PRINT
450 PRINT "KEINE M.",:FOR X=1 TO 3:PRINT C(3,X),:NEXT
```

Beachten Sie, daß am Ende der Zeilen 430 und 440 ein PRINT-Statement stehen muß, um dem Computer mitzuteilen, daß er eine neue Zeile in Position 0 beginnen soll, Auch diese Programmzeilen belegen auf dem Bildschirm übrigens mehr als eine Zeile.

19. Erinnern Sie sich noch an das Problem, die verkauften Schokoladenriegel zu zählen? Vielleicht lesen Sie sich diese Abschnitte in Kapitel 7 noch einmal durch, bevor wir weitergehen?

Damals hatten wir Schokoladenriegel zum Preis von 1 DM verkauft, woran wir einen Verdienst von 55 Pfg. hatten. Jetzt wollen wir noch ein weiteres Produkt hinzunehmen, und zwar eine Tüte mit Geleefrüchten, die wir für 0,50 DM verkaufen. Daran verdienen wir 0,30 DM. Die Aufgabe besteht nun darin, unseren Computer so umzuprogrammieren, daß er die Verkäufe der einzelnen Clubmitglieder und den Gesamt-Verdienst in Form einer Tabelle ausgibt. Hier sehen Sie die Aufstellung, die wir gern haben möchten.

RUN

NUMMER	SUMME	SCHOK.	GEL.FR.
1	0	0	0
2	3	3	0
3	0	0	0
4	6	3	6
5	10	6	8
6	0	0	0
7	0	0	0
8	12	6	12
SUMME	31	18	26
GEWINN:	17.7	9.9	7.8

a) Wer machte in der obigen Aufstellung den größten Umsatz in DM? . .
b) Wer verkaufte die meisten Schokoladenriegel? . . . .

c) Wer verkaufte die meisten Geleefrüchte? ....

d) Können Sie auch sagen, wer uns den größten Gewinn brachte? ....

---

a) 8 b) 5 und 8 c) 8

d) Das ist aus der Aufstellung nicht direkt ersichtlich (es ist Nr. 8).

20. Um die Buchführung zu vereinfachen, bereiten wir ein vorgedrucktes Formular vor, um darauf vermerken zu können, wer was mitgenommen hat. Wenn jetzt eins der Mitglieder Schokolade holt, füllen wir das Formular aus, das etwa so aussehen könnte:

Name ... Danny ...
Nummer ... 4 ...
1. Schokolade ... 3 ...
2. Gelee-Früchte 6 ..

Dieses Formular gibt uns an, daß Danny, der die Nummer 4 hat 3 Schokoladenriegel und 6 Tüten mit Geleefrüchten mitgenommen hat. Um die Zahlenangaben in Daten für den Computer aufzubereiten, benutzen wir folgendes Format:

Nummer, Schokolade oder Gel.Fr. (1 oder 2), Anzahl

Um die Daten in dieser Form darzustellen, werden zwei Datensätze mit insgesamt 6 Zahlenangaben erforderlich:



Vervollständigen Sie bitte, an Hand der drei nachfolgenden Datensätze, die DATA-Statements des anschließenden Programm-Segments. Sehen Sie bitte auch Daten-Ende-Flags vor.

Nummer 5	Nummer 2	Nummer 8
1. 6	1. 3	1. 6
2. 8	2. 0	2. 12

```
900 REM***NUMMER' WAREN–NUMMER, MENGE
910 DATA 4,1,3, 4,2,6
920 ..............................................
930 ..............................................
```

```
920 DATA 5,1,6  5,2,8  2,1,3  2,2,0
930 DATA 8,1,6  8,2,12,  –1,–1,–1
```

Der letzte Zahlenwert (0) hätte auch entfallen können, da dieses Mitglied keine Gelee-Früchte verkauft hat.

Daten-Ende-Flag (→ –1,–1,–1)

21. Unser Array wird so aussehen:

Array A	1=Schokolade	2=Geleefrüchte
1= Jerry		
2=Bobby		
3=Mary		
4=Danny		
5=Karl		
6=Mimi		
7=Doug		
8=Scott		

Hier sehen Sie eine weitere Möglichkeit zur Darstellung des gleichen A-Arrays.

A		A	
(1,1)		(1,2)	
(2,1)		(2,2)	
(3,1)		(3,2)	
(4,1)		(4,2)	
(5,1)		(5,2)	
(6,1)		(6,2)	
(7,1)		(7,2)	
(8,1)		(8,2)	

Nummer ↑ ↑ Schok. ↑ Menge   Nummer ↑ ↑ Gel.Fr. ↑ Menge

Unsere erste Aufgabe besteht darin, die Statements zur Initialisierung des Arrays zu schreiben, d.h. die Dimensionen festzulegen und alle Variablen auf Null zu setzen. Vergessen Sie dabei das Null-Element nicht. Wir werden es nämlich auch benötigen!

```
100 REM***SUESSIGKEITEN–ZAEHLER
110 REM***INITIALISIERUNG
120 ..............................................
130 ..............................................
140 ..............................................
150 ..............................................
```

– –

```
120 DIM A(8,2)
130 FOR X=0 TO 8:FOR Y=0 TO 2
140 A(X,Y)=0
150 NEXT Y:NEXT X
```

22. Lesen Sie jetzt einen Datensatz K,C,Q ein (K = Mitgliedsnummer, C = Süßigkeiten-Art, – 1 oder 2 – , Q = Menge) und prüfen Sie, ob es sich bereits um das Daten-Ende-Flag handelt. Sobald dieses Flag vorliegt, soll das Programm zu Zeile 410 springen.

```
200 REM***LIES DATEN UND PRUEFE AUF DAS FLAG
210 ..............................................
```

– –

```
210 READ K,C,Q : IF K=–1 THEN 410
```

23. Der Kern dieses Programms steht in Zeile 220, in der die Aufaddierung erfolgt:

```
220 A(K,C)=A(K,C)+Q : GOTO 210
```

(Pfeil auf 210:) Gehe zurück zur Zeile 210 und lies weitere Daten.

Nehmen wir einmal an, wir haben folgende Daten:

```
910 DATA 4,1,3
```

a) Bei welchem Array-Element wird durch diesen Datensatz der Wert erhöht? ....

b) Um wieviel wird er erhöht? ....

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

a) a(4,1) b) 3

24. Nehmen wir an, das Array sieht folgendermaßen aus:

A		A	
(1,1)	7	(1,2)	10
(2,1)	0	(2,2)	0
(3,1)	6	(3,2)	8
(4,1)	5	(4,2)	12
(5,1)	3	(5,2)	0
(6,1)	6	(6,2)	8
(7,1)	8	(7,2)	9
(8,1)	3	(8,2)	10

Nummer – Schokolade – Menge; Nummer – Gelee-Früchte – Menge

Wie wird das Array nach der Verarbeitung der folgenden DATA-Statements aussehen?

```
910  DATA 4,1,3,  4,2,6
920  DATA 5,1,6,  5,2,8,  2,1,3,  2,2,0
930  DATA 9,1,6,  8,2,12
```

A		A	
(1,1)		(1,2)	
(2,1)		(2,2)	
(3,1)		(3,2)	
(4,1)		(4,2)	
(5,1)		(5,2)	
(6,1)		(6,2)	
(7,1)		(7,2)	
(8,1)		(8,2)	

- -

A		A	
(1,1)	7	(1,2)	10
(2,1)	3	(2,2)	0
(3,1)	6	(3,2)	8
(4,1)	8	(4,2)	18
(5,1)	9	(5,2)	8
(6,1)	6	(6,2)	8
(7,1)	8	(7,2)	9
(8,1)	9	(8,2)	22

25. Hier ist das Programm, soweit wir es bisher fertiggestellt haben.

```
100 REM***SUESSIGKEITEN–ZAEHLER
110 REM***INITIALISIERUNG
120 DIM A(8,2)
130 FOR X=0 TO 8:FOR Y=0 TO 2
140 A(X,Y)=0
150 NEXT Y:NEXT X
200 REM***LIES DATEN UND PRUEFE AUF DAS FLAG
210 READ K,C,Q:IF K=–1 THEN 310
220 A(K,C)=A(K,C)+Q:GOTO 210
910 DATA 4,1,3, 4,2,6
920 DATA 5,1,6, 5,2,8, 2,1,3, 2,2,0
930 DATA 8,1,6, 8,2,12, –1,–1,–1
```

Wir wollen jetzt einmal eine vorläufige Liste ausdrucken lassen, die der Aufstellung aus Abschnitt 19 entspricht, aber noch keine Summen enthält. Bitte füllen Sie die Leeren Stellen im anschließenden Programm aus. Die vom Programm ausgedruckte Liste sollte folgendermaßen aussehen:

```
RUN
NUMMER          SCHOK.          GEL.FR.
1               0               0
2               3               0
3               0               0
```

4	3	6
5	6	8
6	0	0
6	0	0
8	6	12

```
400 REM***AUSWERTUNG 1
410 PRINT "NUMMER" ................................
420 FOR K=1 TO 8
430 PRINT K, <------------------------ das Komma ist wichtig!
440 FOR C=1 TO ................................
450 PRINT ......................., <- das Komma ist wichtig!
460 NEXT C
470 PRINT
480 NEXT .......................................
```

```
410 PRINT "NUMMER","SCHOK.","GEL.FR."
440 FOR C=1 TO 2
450 PRINT A(K,C),
480 NEXT K
```

26. Warum sind am Ende der Zeilen 430 und 450 in Abschnitt 25 Kommas vorgesehen? ....

Um den Wagenrücklauf nach PRINT zu verhindern.

27. Warum haben wir in Zeile 470 PRINT vorgesehen? ....
....

Um einen Wagenrücklauf zu erzwingen, damit die nächste Mitgliedsnummer, bei erneuter Ausführung der Zeile 420, an der gleichen Stelle ausgedruckt wird.

28. Bis jetzt liefert unser Programm die in Abschnitt 25 gezeigte Aufstellung. Wir wollen es jetzt jedoch so vervollständigen, daß es die in Abschnitt 19 gezeigte Tabelle erstellt.

Dazu verwenden wir die bisher nicht benutzten 0,0-Elemente unseres Arrays.

Die verkauften Stückzahlen werden hier addiert.

(0,0) (1,0) (2,0) (3,0) (4,0) (5,0) (6,0) (7,0) (8,0)

(0,1)

(0,2)

Die von jedem Clubmitglied verkauften Gesamtstückzahlen werden in der ersten Spalte eingetragen.

Füllen Sie bitte die leeren Stellen im nachfolgenden Programm aus. Zeile 130 setzt alle Array-Elemente auf 0, beginnend mit dem 0,0-Element. Zeile 320 addiert die Gesamt-Umsätze jedes Clubmitglieds, Zeile 350 schließlich addiert die verkaufte Stückzahl.

```
100 REM***SUESSIGKEITEN–ZAEHLER
110 REM***INITIALISIERUNG
120 DIM A(8,2)
130 FOR X=0 TO 8:FOR Y=0 TO 2
140 A(X,Y)=0
150 NEXT Y:NEXT X
200 REM***LIES DATEN UND PRUEFE AUF DAS FLAG
210 READ K,C,Q:IF K=–1 THEN 310
220 A(K,C)=A(K,C)+Q:GOTO 210
300 REM***VERKAUFTE ANZAHL UND
301 REM***GESAMTUMSATZ IN DM
310 FOR K=1 TO .................................
320 A(K,0)=A(K,1)*1.00+A(K,2)*.50
330 A(0,0)=A(0,0)+A(K,0)
```

```
340 FOR C= ..........................................
350 A(0,C)=A(0,C)+A(K,...................)
360 NEXT C : .........................................
370 REM***BERECHNE DEN VERDIENST
380 P1=A(0,1)*.55:P2= ...............................
390 P=P1+P2
```

```
310 FOR K=1 TO 8
340 FOR C=1 TO 2
350 A(0,C)=A(0,C)+A(K,C)
360 NEXT C : NEXT K
380 P1=A(0,1)*.55 : P2=A(0,2)*.30
```

29. Haben Sie verstanden, was in den Zeilen 320, 330 und 350 geschieht? Wenn nicht, lesen Sie alles nochmals gründlich durch und beschäftigen Sie sich mit diesem Problem. Diese Statements sind von entscheidender Bedeutung für das Verständnis dieser Art von Probleme.

Tragen Sie in der folgenden Abbildung, in den bezeichneten Feldern, diejenigen Zeilennummern ein, die Werte für diese Felder erzeugen.

(0,0)		(0,1)		(0,2)	
(1,0)					
(2,0)					
(3,0)					
(4,0)					
(5,0)					
(6,0)					
(7,0)					
(8,0)					

(0,0)	330	(0,1)	350	(0,2)	350
(1,0)	320				
(2,0)	320				
(3,0)	320				
(4,0)	320				
(5,0)	320				
(6,0)	320				
(7,0)	320				
(8,0)	320				

30. Was beweirkt Zeile 330 in Abschnitt 28? . . . .

. . . .

330 A(0,0)=A(0,0)+A(k,0)

— —

akkumuliert die Gesamt-Umsätze in A(0,0)

31. Versuchen Sie zu beschreiben, was das Statement P=P1=P2 in Zeile 390 bewirkt. . . . .

. . . .

390 P=P1+P2

— —

Es berechnet den Gesamtgewinn und weist ihn der Variablen P zu

32. Jetzt können wir den Abschnitt unseres Programms, der für den Ausdruck sorgt, auf den letzten Stand bringen. Bitte füllen Sie dazu die leeren Stellem im folgenden Programm aus:

RUN

NUMMER	SUMME	SCHOK.	GEL.FR.
1	0	0	0
2	3	3	0
3	0	0	0
4	6	3	6
5	10	6	8
6	0	0	0

7	0	0	0
8	12	6	12
SUMMEN:	31	18	26
GEWINNE:	17.7	9.9	7.8

```
400 REM***DRUCKE LISTE
410 PRINT "NUMMER" ....................................
420 FOR K=1 TO 8
430 PRINT .............................................
440 FOR C= ............................................
450 PRINT A ...........................................
460 NEXT C : ..........................................
470 NEXT X
480 PRINT "SUMMEN:",A(0,0),A(0,1),A(0,2)
490 PRINT "GEWINNE:" ..................................
```

– –

```
400 REM***DRUCKE LISTE
410 PRINT "NUMMER","SUMME","SCHOK.","GEL.FR."
420 FOR K=1 TO 8
430 PRINT K,
440 FOR C=0 TO 2
450 PRINT A(K,C),
460 NEXT C:PRINT
470 NEXT K
480 PRINT "SUMMEN:",A(0,0),A(0,1),A(0,2)
490 PRINT : PRINT "GEWINNE:",P,P1,P2
```

Hier finden Sie jetzt abschließend das Listing für das komplette Programm.

```
100 REM***SUESSIGKEITEN–ZAEHLER
110 REM***INITIALISIERE
120 DIM A(8,2)
130 FOR X=0 TO 8:FOR Y=0 TO 2
140 A(X,Y)=0
150 NEXT Y;NEXT X
200 REM***LIES DATEN UND PRUEFE AUF DAS FLAG
210 READ K,C,Q:IF K=–1 THEN 310
220 A(K,C)=A(K,C)+Q:GOTO 210
```

```
300 REM***GESAMTSTUECKZAHL UND
301 REM***GESAMTUMSATZ IN DM
310 FOR K=1 TO 8
320 A(K,0)=A(K,1)*1+A(K,2)*0.5
330 A(0,0)=A(0,0)+A(K,0)
340 FOR C=1 TO 2
350 A(0,C)=A(0,C)+A(K,C)
360 NEXT C:NEXT K
370 REM***BERECHNE DIE GEWINNE
380 P1=A(0,1)*0.55:P2=A(0,2)*0.3
390 P=P1+P2
400 REM***PRINT LISTE
410 PRINT "NUMMER","SUMME","SCHOK.","GEL.FR."
420 FOR K=1 TO 8
430 PRINT K,
440 FOR C=0 TO 2
450 PRINT A(K,C),
460 NEXT C:PRINT
470 NEXT K
480 PRINT "SUMMEN:",A(0,0),A(0,1),A(0,2)
490 PRINT :PRINT "GEWINNE:",P,P1,P2
910 DATA 4,1,3, 4,2,6
920 DATA 5,1,6, 5,2,8, 2,1,3, 2,2,0
930 DATA 8,1,6, 8,2,12,-1,-1,-1
```

32. Nun noch eine letzte Anwendung für doppelt-indizierte Arrays: In einer kleinen Klasse mit 8 Schülern hat jeder an vier Klassenarbeiten teilgenommen und dabei die nachfolgenden Punktzahlen erreicht.

	Arbeit 1	Arbeit 2	Arbeit 3	Arbeit 4
Schüler 1	65	57	71	75
Schüler 2	80	90	91	88
Schüler 3	78	82	77	86
Schüler 4	45	38	44	46
Schüler 5	83	82	79	85
Schüler 6	70	68	83	59
Schüler 7	98	92	100	97
Schüler 8	85	73	80	77

S(B,J) soll die Punktzahl sein, die von Schüler B in der Klassenarbeit J erreicht wurde. S(5,2) ist dann die Punktzahl, die Schüler .... in der Arbeit ...... erzielte. Wie groß ist der Wert von S(5,2)? ......

5; 2; 82

33. In einer anderen Klasse könnten beispielsweise 30 Schüler sitzen, von denen jeder 5 Klassenarbeiten geschrieben hat. Noch eine weitere Klasse hat vielleicht 23 Schüler mit je 7 Klassenarbeiten, usw. Wir wollen daher ein Programm schreiben, um die Punktzahlen von N Schülern bei Q Klassenarbeiten/Schüler einzulesen.

```
100 REM***KLASSENARBEIT–AUSWERTUNGSPROGRAMM
110 DIM S(30,10)
```

Das DIM-Statement ist für bis zu ..... Schülern und maximal ...... Klassenarbeiten geeignet.

30; 10

34. Als nächstes wollen wir die Werte von N und Q für die in Abschnitt 32 abgebildete Punkte-Tabelle einlesen. Für diese Daten ist der Wert von N (Anzahl der Schüler) .... und der Wert von Q (Anzahl der Klassenarbeiten) ....

8; 4

35. Wir setzen die Werte für N und Q, sowie die Punktzahlen in DATA-Statements ein. Das Programm sieht dann folgendermaßen aus:

```
100 REM***KLASSENARBEIT–AUSWERTUNGSPROGRAMM
110 DIM S(30,10)
900 REM***ZEILE 910: WERTE VON N UND Q
910 DATA 8,4 <-------------------- Werte von N und Q
919 REM***LIES DIE PUNKTZAHLEN
```

```
920 DATA 65,57,71,75
930 DATA 80,90,91,88
940 DATA 78,82,77,86
950 DATA 45,38,44,46
960 DATA 83,82,79,85
970 DATA 70,68,83,59
980 DATA 98,92,100,97
990 DATA 85,73,80,77
```

(N x Q) – Array der Punktezahlen aus Abschnitt 32

Jetzt sind Sie daran! Vervollständigen Sie bitte die nachfolgende Zeile 120, mit der die Werte für N und Q eingelesen werden.

120 ....

— — — — — — — — — — — — — — — — — —

120 READ N,Q

36. Die durch Zeile 120 in Abschnitt 35 gelesenen Werte für N und Q werden aus welchem DATA-Statement gelesen? ....

— — — — — — — — — — — — — — — — — —

Zeile 910

37. Jetzt wollen wir das NxQ-Array der Punktzahlen, unter Verwendung von FOR-NEXT-Schleifen, einlesen. Füllen Sie dazu bitte die leeren Stellen in den nachfolgenden Statements aus.

130 FOR X=1 TO N : FOR Y= .... :READ .... NEXT Y:NEXT X

— — — — — — — — — — — — — — — — — —

130 FOR X=1 TO N:FOR Y=1 TO Q:READ S(X,Y):NEXT Y:NEXT X

38. Die durch Zeile 130 gelesenen numerischen Werte sind in den DATA-Statements von Zeile .... bis ...... gespeichert.

— — — — — — — — — — — — — — — — — —

920; 990

39. Nachdem wir das Array nun im Computer haben, – was fangen wir damit an? Beispielsweise könnte uns die mittlere Punktzahl eines jeden Schülers interessieren. Fangen wir damit einmal an, beginnend mit Zeile 200.

```
200 REM***BERECHNE UND DRUCKE DIE
201 REM***DURCHSCHNITTSWERTE
210 PRINT "SCHUELER","DURSCHNITT"
220 FOR B=1 TO N        ┐
230 LET T=0             │
240 FOR J=1 TO Q        ├─Berechne die Summe der Punkte für Schüler B
250 LET T=T+S(B,J)      │
260 NEXT J              ┘
270 LET A=T/Q ◄──────────── Durchschnittswert von Schüler B
280 PRINT B,A ◄──────────── Schülernummer und Durchschnitt
290 NEXT B ◄─────────────── Berechne die Werte für den nächsten Schüler
```

Die Zeilen 230 bis 280 werden für jeden Schüler ausgeführt. Das heißt, zunächst für B=1, dann für B=2 usw., bis zu B=N. Welcher Wert für T wird für B=1 durch die Zeilen 230 bis 260 berechnet? T= .... ?

– –

268. Das ist die Summe der vier Punktzahlen von Schüler 1. Erinnern Sie sich: Q ist 4. Daher wird Zeile 250 für J=1, J=2, J=3 und J=4 ausgeführt.

40. Welcher Wert von A wird für B=1 durch Zeile 270 berechnet? A=T/Q= ....

– –

67; A=T/Q=268/4=67

41. Hier nun das vollständige Programm und ein Lauf.

```
100 REM***KLASSENARBEIT–AUSWERTUNGSPROGRAMM
110 DIM S(30,10)
120 READ N,Q
130 FOR X=1 TO N:FOR Y=1 TO Q
140 READ S:S(X,Y)=S
150 NEXT Y:NEXT X
200 REM***BERECHNE UND DRUCKE DIE
201 REM***DURCHSCHNITTSWERTE
210 PRINT "SCHUELER","DURCHSCHNITT"
220 FOR B=1 TO N
230 T=0
```

```
240 FOR J=1 TO Q
250 T=T+S(B,J)
260 NEXT J
270 A=T/Q
280 PRINT B,A
290 NEXT B
900 REM***ZEILE 910:WERTE VON N UND Q
910 DATA 8,4
919 REM***PUNKTZAHLEN
920 DATA 65,57,71,75
930 DATA 80,90,91,88
940 DATA 78,82,77,86
950 DATA 45,38,44,46
960 DATA 83,82,79.85
970 DATA 70,68,83,59
980 DATA 98,92,100,97
990 DATA 85,73,80,77
```

```
RUN
SCHUELER       DURCHSCHNITT
1              67
2              87.25
3              80.75
4              43.25
5              82.25
6              70
7              96.75
8              78.75
```

Jetzt sollen Sie, beginnend mit Zeile 300, ein Programmsegment schreiben, das die durchschnittliche Punktzahl für jede Klassenarbeit berechnet und ausdruckt. Für die in diesem Programm verwendeten Daten sehen die Ergebnisse fogendermaßen aus:

```
RUN
ARBEIT         DURCHSCHNITT
1              75.5
2              72.75
3              78.125
4              76.625
```

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

\- -

```
300 REM***BERECHNE UND DRUCKE DEN DURCHSCHNITT
301 REM***JEDER ARBEIT
310 PRINT "ARBEIT", "DURCHSCHNITT"
320 FOR J=1 TO Q
330 LET T=0
340 FOR B=1 TO N
350 LET T=T+S(B,J)
360 NEXT B
370 LET A=T/N
380 PRINT J,A
390 NEXT J
```

42. Nehmen wir einmal an, 10 Studenten nehmen an einem Multiple-Choice-Test teil, der 10 Fragen mit jeweils vier möglichen Antworten je Frage umfaßt. Wir möchten jetzt gern wissen, wieviele Studenten bei Frage 1 die Antwort 1, wieviele die Antwort 2 usw. gaben.
Hier sehen Sie die von 7 Studenten gegebenen Antworten. Für jeden Antworten-Satz ist ein DATA-Statement vorgesehen. Das letzte Statement gilt für einen "fiktiven" Studenten und dient lediglich als Daten-Ende-Flag.

```
900 REM***ANTWORTEN DER STUDENTEN
910 DATA 2,3,1,1,1,2,4,3,4,1
920 DATA 2,3,2,4,1,2,4,2,1,1
930 DATA 2,3,2,4,1,2,4,2,1,1
940 DATA 3,2,4,1,1,2,3,3,4,1
950 DATA 2,3,4,1,1,3,4,3,4,1
960 DATA 2,1,2,3,1,2,4,3,4,2
```

```
970 DATA 3,4,1,1,1,4,3,1,4,2
980 DATA -1,0,0,0,0,0,0,0,0,0          "fiktiver Student"
```

In jeder Datenzeile ist die erste Zahl die Antwort auf Frage 1, die zweite die Antwort auf Frage 2 usw. Student 1 (Zeile 910) gab bei frage 3 die Antwort 1. Student 5 (Zeile 950) gab auf Frage 9 die Antwort . . . . ., Student 7 (Zeile 970) gab Antwort . . . . . auf Frage 1.

— — — — — — — — — — — — — — — — — — —

4; 3

43. Bitte vervollständigen Sie die folgende Tabelle. Sie gibt für jede Frage an, von wievielen Studenten die einzelnen möglichen Antworten gewählt wurden.

	Antwort 1	Antwort 2	Antwort 3	Antwort 4
Frage 1	0	5	2	0
Frage 2	1	1	. . . . . .	. . . .
Frage 3	. . . . . .	. . . . . .	. . . . . .	. . . .

— —

4 1

2 3 0 2

44. In Abschnitt 43 haben wir, mit Ihrer Unterstützung, gezeigt wie die 7 Studenten die ersten drei Fragen beantwortet haben. Die Ergebnisse sind angeordnet wie ein 3x4-Array. Wenn wir diese Tabelle für sämtliche 10 Fragen fortgesetzt hätten, würde sie aussehen wie ein . . . . x 4 Array.

— — — — — — — — — — — — — — — — — — — —

10

45. Wir wollen jetzt ein Array T mit 10 Reihen und 4 Spalten definieren, das die Anzahl der für jede Frage gegebenen Antworten (wie in der Tabelle in Abschnitt 43) aufnimmt. Bitte vervollständigen Sie das nachfolgende DIM-Statement.

```
100 REM***TEST-ANALYSE-PROGRAMM
110 DIM ..........................................
```

— — — — — — — — — — — — — — — — — — — —

110 DIM T(10,4)

46. Von jedem Studenten wurden 10 Antworten gegeben. Wir wollen daher noch eine Antworten-Liste A(1) bis A(10) definieren. Bitte vervollständigen Sie auch dieses DIM-Statement:

120 DIM ....

---

120 DIM A(10)

47. Wir können Speicherplatz sparen, wenn wir beide DIM-Statements zu einem zusammenfassen.

110 DIM T(10,4),A(10)

Dieses DIM-Statement definiert ein ...... mit der Bezeichnung T mit maximal 10 Reihen und 4 Spalten, sowie ein .... mit der Bezeichnung A und maximal 10 Elementen.

---

zweidimensionales Array, Matrix oder Tabelle
eindimensionales Array, Liste oder Vektor

Achten Sie darauf, daß ein Komma zur Trennung von T(10,4) und A(10) benötigt wird.

48. Hier ist der Anfang eines Programms, durch das die Antworten der Studenten gelesen und das Array T berechnet wird.

```
100 REM***TEST–ANALYSE–PROGRAMM
110 DIM T(10,4),A(10)
```

Als nächstes müssen wir das T-Array, das die Anzahl der abgegebenen Antworten für jede Frage enthält, initialisieren. Das heißt bekanntlich, daß wir alle Array-Elemente auf Null setzen.

Bitte führen Sie die entsprechenden Statements aus.

```
120 REM***INITIALISIERUNG
130 ..........................................
140 ..........................................
```

150 ....

```
120 REM***INITIALISIERUNG
130 FOR X=1 TO 10:FOR Y=1 TO 4
140 T(X,Y)=0
150 NEXT Y:NEXT X
```

49. Schreiben Sie bitte ein Statement zum Lesen der Liste A (Antworten eines Studenten):

```
200 REM***LIES EINEN ANTWORTENSATZ
210 ..............................................
220 ..............................................
230 ..............................................
```

```
200 REM***LIES EINEN ANTWORTENSATZ
210 FOR X=1 TO 10
220 READ A:A(X)=A
230 NEXT X
```

50. Natürlich müssen wir bei jedem Lesen auch prüfen, ob wir evtl. bereits am Daten-Ende-Flag angelangt sind. Sie erinnern sich sicherlich daran, daß wir am Ende der Daten einen "fiktiven" Studenten vorgesehen hatten. Wenn das Programm bei diesem "Datensatz" angelangt ist, sollen die Antworten ausgedruckt werden, und zwar beginnend mit Zeile 510. Vervollständigen Sie zunächst das folgende IF-THEN-Statement.

```
140 REM***PRUEFEN AUF DAS DATENENDE
250 IF ..................... THEN 510
```

A(1)=–1

51. Solange noch wirkliche Daten eingelesen werden, soll der laufende Zähler im T-Array ständig auf den neuesten Stand gebracht werden. Wir haben diese Aufgabe folgendermaßen gelöst:

```
300 REM***SUMMEN IM T–ARRAY
```

```
310 FOR Q=1 TO 10
320 T(Q,A(Q))=T(Q,A(Q))+1
330 NEXT Q
```

Hier sehen Sie die Antworten eines Studenten. Es handelt sich dabei um die Werte für A(1) bis A(10).

2,3,1,1,1,2,4,3,4,1

Nehmen Sie an, Q ist 1. Dann ist A(Q)= .... und T(Q,A(Q)) ist T(..........................).

– –

2; T(1,2) (da Q=1 und A(Q)=2)

52. Was geschieht im obigen Programmsegment, wenn der Computer Zeile 320 mit den angegebenen Daten ausführt? ....

– –

Die Summe in T(1,2) wird um 1 erhöht.

53. Da die Zeile 320 in einer FOR-NEXT-Schleife liegt, wird sie für jeden Wert von Q ausgeführt, der durch das FOR-Statement spezifiziert ist, das heißt für Q=1,2,3,4,5,6,7,8,9 und 10. Welches Element der T-Matrix wird für Q=10 um 1 erhöht? T(....................)

– –

T(10,A(10)) oder T(10,1) für die Daten in Abschnitt 51.

54. Machen wir weiter! Nachdem wir die Antworten eines Studenten aufaddiert haben, möchten wir, daß der Computer zur Zeile 210 zurückgeht und einen weiteren Antwortsatz einliest (siehe Abschnitt 49).

```
400 REM***EIN WEITERER ANWORTENSATZ
410 GOTO 210
```

Dann wird das IF-THEN-Statement erneut ausgeführt. Es veranlaßt den Computer, zur Zeile 510 zu springen, sobald der Datensatz des "fiktiven" Studenten gelesen wird. In dem Fall möchten wir, daß die Überschriften und Ergebnisse ausgedruckt werden und der Computer anschließend anhält.

Ein Lauf sollte dann so aussehen:

	A1	A2	A3	A4
S 1	0	5	2	0
S 2	1	1	4	1
S 3	2	3	0	2
S 4	4	0	1	2
S 5	7	0	0	0
S 6	0	5	1	1
S 7	0	0	2	5
S 8	1	2	4	0
S 9	2	0	0	5
S10	5	2	0	0

Vervollständigen Sie bitte diesen Programmabschnitt:

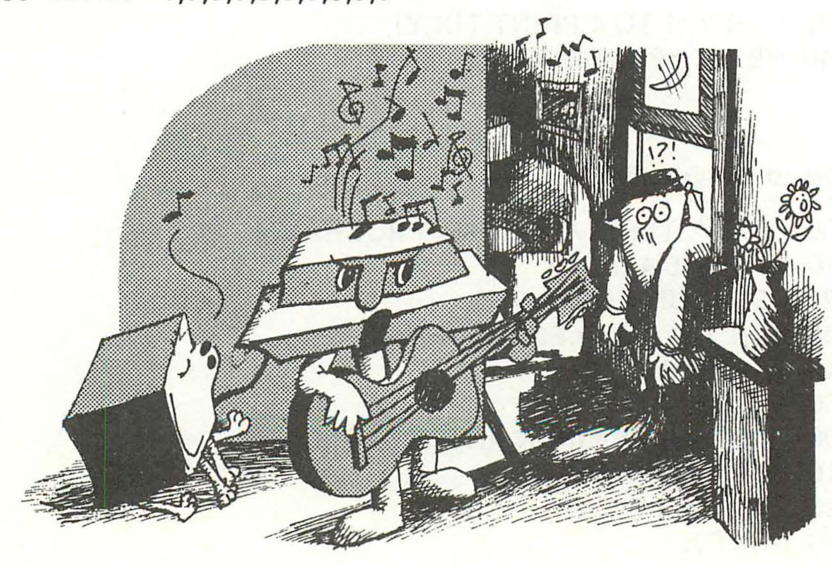
```
500 REM***DRUCKE DAS T-ARRAY
501 REM***A=NUMMER DER ANTWORT
502 REM***S=NUMMER DES STUDENTEN
510 PRINT "  A 1","  A 2","  A 3","  A 4"
520 FOR X=1 TO 10:PRINT "S";X;" ";
530 ..............................................
540 ..............................................
```

```
530 FOR Y=1 TO 4:PRINT T(X,Y),
540 NEXT Y:PRINT:NEXT X
```

Hier ist das vollständige Listing des Programms:

```
100 REM***TEST-ANALYSE-PROGRAMM
110 DIM T(10,4),A(10)
120 REM***INITIALISIERE
130 FOR X=1 TO 10:FOR Y=1 TO 4
140 T(X,Y) =0
150 NEXT Y:NEXT X
200 REM***LIES EINEN ANTWORTENSATZ
210 FOR X=1 TO 10
220 READ A:A(X)=A
230 NEXT X
```

```
240 REM***PRUEFEN AUF DAS DATENENDE
250 IF A(1)=-1 THEN 510
300 REM***SUMMEN IM T-ARRAY
310 FOR Q=1 TO 10
320 T(Q,A(Q))=T(Q,A(Q))+1
330 NEXT Q
400 REM***EIN WEITERER ANTWORTENSATZ
410 GOTO 210
500 REM***DRUCKE DAS T-ARRAY
501 REM***A=NUMMER DER ANTWORT
502 REM***S=NUMMER DER STUDENTEN
510 PRINT "  A 1","  A 2","  A 3","  A 4"
520 FOR X=1 TO 10:PRINT "S",X;" ";
530 FOR Y=1 TO 4:PRINT T(X,Y),
540 NEXT Y:PRINT:NEXT X
900 REM***ANTWORTEN DER STUDENTEN
910 DATA 2,3,1,1,1,2,4,3,4,1
920 DATA 2,3,2,4,1,2,4,2,1,1
930 DATA 2,3,2,4,1,2,4,2,1,1
940 DATA 3,2,4,1,1,2,3,3,4,1
950 DATA 2,3,4,1,1,3,4,3,4,1
960 DATA 2,1,2,3,1,2,4,3,4,2
970 DATA 3,4,1,1,1,4,3,1,4,2
980 DATA -1,0,0,0,0,0,0,0,0,0
```



EIGENTEST

Die folgenden Aufgaben sollen Ihnen helfen, noch einmal die Anwendung aller BASIC-Instruktionen zu üben, die etwas mit Zahlen-Arrays zu tun haben, die doppelt-indizierte Variable enthalten.

1. Kreuzen Sie an, welche der nachfolgenden, doppelt-indizierten Variablen in BASIC gültig sind.

   ...... a) X(2+2) ...... b) X(5,5)
   ...... c) X1(100,100) ...... d) X(A+B,C)
   ...... e) X(X(1,2),X(2,1) ...... f) X(A,A)

Die Fragen 2 bis 7 bezeihen sich auf das folgende Array:

	Spalte 1	Spalte 2
Reihe 1	1	2
Reihe 2	3	4
Reihe 3	5	6

2. Welche Dimensionen hat A? ....

3. Schreiben Sie für A ein DIM-Statement.

   100 ....

4. Welche Variable lokalisiert die Position in der dritten Reihe und zweiten Spalte von A? ....

5. Wie lautet der Wert der folgenden Array-Elemente?

   a) A(1,1) .... b) A(3,1) ....

6. X Soll 2 und Y=3 sein. Wie lautet dann der Wert der folgenden Array-Elemente?

   a) A(X,X) .... b) A(X+1,Y−1) ....

7. Wie groß ist der Wert von A(A(1,2),A(2,1)−1)? ....

8. Schreiben Sie ein Programm, das zwei FOR-NEXT-Schleifen verwendet, um ein 10x10-Array (M) mit Nullen zu füllen.

   ....

   ....

....

....

....

....

9. Für Ihren Stadtteil soll eine Einwohnerstatistik erstellt werden. Unter anderem möchte man gern die Altersverteilung und die Anzahl der männlichen und weiblichen Einwohner ermitteln. Man bittet Sie, Ihren Heimcomputer zu programmieren, um die mit dem nachfolgenden Formular erfaßten Daten zu verarbeiten und eine Übersicht zu erstellen, die dem am Schluß angegebenen Muster entspricht.

Hier zunächst einmal das Erhebungsformular.

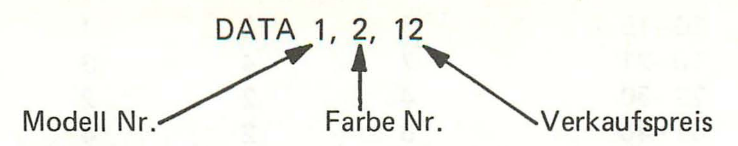
EINWOHNERBEFRAGUNG (Bitte kreuzen Sie ein Feld für jede Frage an)	
Frage 1: Altersgruppe ☐ 1. jünger als 10 ☐ 2. 10 – 15 ☐ 3. 16 – 21 ☐ 4. 22 – 30 ☐ 5. 31 – 40 ☐ 6. 41 – 50 ☐ 7. 51 – 65 ☐ 8. über 65	Frage 2: Geschlecht ☐ 1. männlich ☐ 2. weiblich

Verwenden Sie in Ihrem Programm die folgenden DATA-Statements.

```
910 REM***DATEN: ALTERSGRUPPE–GESCHLECHT
910 DATA 1,2,2,1,3,2,4,1,5,2,6,2,7,2
920 DATA 8,1,1,2,1,1,2,1,2,2,3,1,3,1
930 DATA 3,2,3,1,3,2,4,1,4,2,5,1,5,1
940 DATA 5,2,5,2,6.1,6,2,7,1,7,2,7,2
950 DATA 7,2,8,1,8,1,8,2,8,2,3,1,4,2
960 DATA 5,2,6,1,6,1,7,1,8,2,–1,–1
969 REM***TABELLEN–UEBERSCHRIFTEN
970 DATA < 10,10–15,16–21,22–30,31–40,41–50,51–65, > 65
RUN
ALTER          SUMME    MAENNL.   WEIBL.
< 10           3        1         2
```

10–15	3	2	1
16–21	7	4	3
22–30	4	2	2
31–40	6	2	4
41–50	5	3	2
51–65	6	2	4
>65	6	3	3
SUMMEN	40	19	21

10. Ihr Schwager ist Besitzer eines Jeans-Geschäfts. Er hat ein sehr großes Angebot mit einer umfangreichen Auswahl, möchte seinen Bestand aber etwas reduzieren. Die Frage ist nur, bei welchen Artikeln er damit anfangen soll. Daher bittet er Sie, Ihren Computer dazu zu benutzen, seine Verkäufe zu analysieren und ihm bei der Entscheidung zu helfen, welche Artikel er auswählen soll. Als erstes möchte er gern die Artikelgruppe "Kinderhosen" untersuchen. Er führt 9 verschiedene Modelle, jedes in drei Farben. Für Ihr Programm sollten Sie die Modelle mit 1 bis 9 und die Farben mit 1 bis 3 numerieren, sodaß jeder Datensatz folgendermaßen aussieht:



Schreiben Sie bitte ein Programm zur Verkaufsanalyse und bereiten Sie eine Aufstellung vor, die so aussieht wie das am Schluß der Aufgabe abgebildete Beispiel.

In Ihrem Programm verwenden Sie bitte die folgenden DATA-Statements.

```
900 REM***MODELL,FARBE,UMSATZ IN DM
910 DATA 1,1,12, 1,2,13, 1,3,14
911 DATA 2,1,22, 2,2,10, 2,3,12
912 DATA 3,1,45, 3,2,14, 3,3,32
913 DATA 4,1,13, 4,2,10, 4,3,15
914 DATA 5,1,12, 5,2,13, 5,3,12
915 DATA 6,1,12, 6,2,12, 6,3,12
916 DATA 7,1,12, 7,2,12, 7,3,12
917 DATA 8,1,14, 8,2,14, 8,3,14
918 DATA 9,1,15, 9,2,15, 9,3,15
919 DATA 3,2,35, 5,1,50, 6,1,36
920 DATA 9,3,12, 2,3,40, 5,1,24
921 DATA 3,3,14, 4,1,48, 1,1,12
922 DATA 2,1,12, 3,1,13, –1,–1,–1
```

Hier ist der Teil des Programms, der die Tabelle ausdruckt. Schreiben Sie bitte den Rest.

```
300 REM***DRUCKE TABELLE
310 PRINT "S    TOT. DM","FARBE 1","FARBE 2","FARBE 3"
320 FOR S=1 TO 9
330 PRINT S;"  ";T(S,0),
340 FOR C=1 TO 3
350 PRINT T(S,C),
360 NEXT C:PRINT
370 NEXT S
380 FOR X=0 TO 3
390 PRINT "T=";T(0,X),
400 NEXT X
```

....

....

RUN

S	TOT. DM	FARBE 1	FARBE 2	FARBE 3
1	51	24	13	14
2	96	34	10	52
3	153	58	49	46
4	66	61	10	15
5	111	86	13	12
6	72	48	12	12
7	39	13	13	13
8	42	14	14	14
9	57	12	15	27
	T=707	T=353	T=149	T=205

Antworten zum Eigentest

Die Zahlen in Klammern geben die Abschnitte an, in denen der jeweilige Stoff behandelt wurde.

1. b), d), e) und f) sind gültig (längere Variablennamen, wie z.B. c) können bei Ihrem Computer evtl. zulässig sein. Bitte überprüfen Sie Ihr Manual daraufhin). (Abschnitte 5 – 7)

2. 3,2 bedeutet: 3 Reihen, 2 Spalten (Abschnitte 2 – 4)

3. 100 DIM A(3,2) (Abschnitte 12)

4. A(3,2) (Abschnitte 9 – 11)

5. a) 1; b) 5 (Abschnitt 11)

6. a) 4; b) 6 (Abschnitte 11)

7. 4 (Abschnitt 11)

8.
```
10 DIM M(10,10)
20 FOR R=1 TO 10
30 FOR C=1 TO 10
40 M(R,C)=0
50 NEXT C
60 NEXT R
```
(Abschnitt 12)

9.

```
100 REM ***BEVOELKERUNGSANALYSE
110 REM ***INITIALISIERE
120 DIM V(8,2),A$(6)
130 FOR A=0 TO 8:FOR S=0 TO 2
140 V(A,S)=0
150 NEXT S:NEXT A
200 REM ***LIES DATEN, PRUEFE, AKKUMULIERE
210 READ A,S:IF A=-1 THEN 310
220 V(A,S)=V(A,S)+1:V(0,0)=V(0,0)+1
230 V(0,S)=V(0,S)+1:V(A,0)=V(A,0)+1
240 GOTO 210
300 REM ***DRUCKE LISTE
```

```
310 PRINT "ALTER","SUMME","MAENNL.","WEIBL."
320 FOR A=1 TO 8
330 READ A$:PRINT A$,V(A,0),
340 FOR S=1 TO 2
350 PRINT V(A,S),
360 NEXT S:PRINT :NEXT A:PRINT
370 PRINT "SUMMEN",V(0,0),V(0,1),V(0,2)
900 REM ***DATEN: ALTERSGRUPPE-GESCHLECHT
910 DATA 1,2,2,1,3,2,4,1,5,2,6,2,7,2
920 DATA 8,1,1,2,1,1,2,1,2,2,3,1,3,1
930 DATA 3,2,3,1,3,2,4,1,4,2,5,1,5,1
940 DATA 5,2,5,2,6,1,6,2,7,1,7,2,7,2
950 DATA 7,2,8,1,8,1,8,2,8,2,3,1,4,2
960 DATA 5,2,6,1,6,1,7,1,8,2,-1,-1
969 REM ***TABELLEN-UEBERSCHRIFTEN
970 DATA <10,10-15,16-21,22-30,31-40,41-50,
   51-65,>65
```

(Abschnitt 19)

10.

```
100 REM ***VERKAUFS-ANALYSE
110 REM ***INITIALISIERE
120 DIM T(9,3)
130 FOR S=0 TO 9:FOR C=0 TO 3
140 T(S,C)=0
150 NEXT C:NEXT S
200 REM ***LIES, PRUEFE UND AKKUMULIERE
210 READ S,C,D:IF S=-1 THEN 310
220 T(S,C)=T(S,C)+D
230 T(0,0)=T(0,0)+D
240 T(0,C)=T(0,C)+D
250 T(S,0)=T(S,0)+D
260 GOTO 210
300 REM ***DRUCKE TABELLE
310 PRINT "S  DM","FARBE 1","FARBE 2","FARBE 3"
320 FOR S=1 TO 9
330 PRINT S;" ";T(S,0),
340 FOR C=1 TO 3
350 PRINT T(S,C),
360 NEXT C:PRINT
370 NEXT S
380 FOR X=0 TO 3
```

```
390 PRINT "T=";T(0,X),
400 NEXT X
900 REM ***MODELL, FARBE, UMSATZ IN DM
910 DATA 1,1,12, 1,2,13, 1,3,14
911 DATA 2,1,22, 2,2,10, 2,3,12
912 DATA 3,1,45, 3,2,14, 3,3,32
913 DATA 4,1,13, 4,2,10, 4,3,15
914 DATA 5,1,12, 5,2,13, 5,3,12
915 DATA 6,1,12, 6,2,12, 6,3,12
916 DATA 7,1,13, 7,2,13, 7,3,13
917 DATA 8,1,14, 8,2,14, 8,3,14
918 DATA 9,1,15, 9,2,15, 9,3,15
919 DATA 3,2,35, 5,1,50, 6,1,36
920 DATA 9,3,12, 2,3,40, 5,1,24
921 DATA 3,3,14, 4,1,48, 1,1,12
922 DATA 2,1,12, 3,1,13, -1,-1,-1
```

(Abschnitt 32)

KAPITEL 9

String-Variable und String-Funktionen

In Kapitel 3 zeigten wir Ihnen bereits einige Möglichkeiten zur Verwendung alphanumerischer String-Variablen. Aber auch in den folgenden Kapiteln haben wir, bei der Einführung neuer Programmier-Konzepte, String-Variable verwendet. In diesem Kapitel nun wollen wir noch einmal wiederholen, was Sie über String-Variable bereits wissen und Ihnen gleichzeitig natürlich auch noch eine Menge neuer Kenntnisse vermitteln. Sie werden dabei feststellen, daß für String-Variable ganau so viele Manipulationsmaöglichkeiten wie für eine numerische Variable bestehen.

Wenn Sie dieses Kapitel durchgearbeitet haben, können Sie das RESTORE-Statement in Zusammenhang mit den Statements READ und DATA benutzen. Außerdem werden Sie in der Lage sein, die folgenden String-Funktionen in Verbindung mit String-Variablen anzuwenden, von denen Sie in diesem Kapitel noch gneügend erfahren werden:

VAL
STR$
CHR$
ASC
LEN

1. Bis jetzt war für uns die Verwendung alphanumerischer Zeichenketten (Buchstaben und Zahlen gemischt) im wesentlichen auf Strings in PRINT-Statements beschränkt, wie z.B. im folgenden Statement:

```
10 PRINT "DIES IST EIN STRING"
```

Jetzt wollen wir jedoch die String-Variable neu einführen.

```
 5 DIM T$(50)
10 LET T$="STRING FUER DIE STRING–VARIABLE T$"
       ↑
       Dies ist eine String-Variable
```

Eine String-Variable wird durch irgend einen beliebigen, aber zulässigen Variablen-Namen, gefolgt von einem Dollarzeichen ($) gekennzeichnet. String-Variable ermöglichen eine einfachere Manimpulation Alphanumerischer Daten. Die Instruktionen für String- und Variable schließen LET, PRINT, INPUT, READ, DATA und IF-THEN, sowie eine Reihe spezieller String-Funktionen ein.

Wie wir bereits in den vergangenen Kapiteln mehrfach darauf hingewiesen haben, müssen Sie am Anfang eines Programms ein DIM-Statement vorsehen. Es wird nur einmal ausgeführt, um dem Computer mitzuteilen, wieviel Speicherplatz er für die im Programm verwendeten String-Variablen bereitstellen muß. Ein String, der einer String-Variablen unter Verwendung einer LET-Anweisung zugeordnet wird, ist auf eine Länge von ungefähr 99 Zeichen beschränkt. Dies ist auf die Beschränkung der Zeilenlänge der Statements zurückzuführen. Wieviele Zeichen enthält die obige String-Variable T$? ....
Haben Sie auch die Zwischenräume mitgezählt?

— — — — — — — — — — — — — — — — — — —

33 Zeichen

2. Wie Sie wissen, können Sie einer String-Variablen unter Verwendung eines INPUT-Statements einen Wert zuweisen. Das Statement fordert den Anwender auf, Werte für eine oder mehrere String-Variable einzugeben.

Beispiel 1:

```
 5 DIM N$(30)
10 PRINT "WIE HEISSEN SIE"; <———— eine String-Eingabe
20 INPUT N$
```

Beispiel 2:

```
 5 DIM N$(30), S$(30)
10 PRINT "IHR NAME IHR STERNZEICHEN"; zwei String-Eingaben
20 INPUT N$,S$
```

Beispiel 3:

```
 5 DIM N$(30),C$(30),S$(30)
```

```
10 PRINT "IHR NAME' WOHNORT UND DAS LAND IN DEM
   SIE LEBEN"; ◄──────────────── drei String-Eingaben
20 INPUT N$,C$,S$
```

Sie können aber im gleichen INPUT-Statement auch numerische und String-Variable zuweisen, wie es das nächste Beispiel zeigt.

Beispiel 4:

```
 5 DIM N$(30)
10 PRINT "WIE HEISSEN SIE UND WIE ALT SIND SIE";
20 INPUT N$,A ◄──────────── Gemischte Eingabe (ein String,
                            eine Zahl).
```

Trifft der Computer bei der Ausführung eines Programms auf ein INPUT-Statement mit mehr als einer Variable, sei es eine String-Variable oder eine numerische Variable, druckt er ein Fragezeichen und wartet darauf, daß der Anwender den entsprechenden String oder den numerischen Wert über die Tastatur eingibt.

Nehmen wir einmal an, wir haben das obige Beispiel in den Computer eingegeben. Wir tippen dann RUN ein und drücken auf RETURN. Das Display zeigt an:

```
RUN
IHR NAME UND IHR STERNZEICHEN?
```

Ich antworte durch Eingabe meines Namens und drücke auf RETURN.

```
RUN
IHR NAME UND STERNZEICHEN? DIETER MUELLER
```

Dann gebe ich mein Sternzeichen ein, das der zweiten Input-Variablen zugeordnet wird.

```
RUN
IHR NAME UND STERNZEICHEN?  DIETER MUELLER
 ? SKORPION
```

Hätte ich beide Antworten gleichzeitig direkt nach dem ersten Fragezeichen gegeben, wären Sie gemeinsam N$ zugeordnet worden. Betätigt man lediglich RETURN, ohne nach dem Fragezeichen etwas

einzugeben, wird der String-Variablen ein "Null-String", also ein String ohne Inhalt zugewiesen.

```
 5 DIM N$(30)
10 PRINT "WIE HEISSEN SIE UND WIE ALT SIND SIE";
20 INPUT N$,A
30 PRINT "NAME:";N$
40 PRINT "ALTER:";A
RUN
```

....

....

....

....

Bitte geben Sie an, was der Computer ausdruckt und wie Sie die gewünschten Informationen eingeben müssen, wenn das Programm läuft.

— — — — — — — — — — — — — — — — — — — —

```
WIE HEISSEN SIE UND WIE ALT SIND SIE? KLAUS MEIER
? 33
NAME: KLAUS MEIER
ALTER: 33
```

3.
```
10 DIM N$(15)
20 PRINT "IHR NAME";
30 INPUT N$
40 PRINT "IHR NAME IST";N$
```

Nehmen wir einmal an, wir lassen dieses Programm laufen.

```
IHR NAME? JERALD RICHARD BROWN
IHR NAME IST JERALD RICHARD
```

Was ist da geschehen? Hat der Computer etwa meinen Nachnamen "vergessen"? Keineswegs. Sehen Sie sich einmal das DIM-Statement an. Da die String-Variable N$ nur zur Aufnahme von 15 Zeichen dimensioniert worden war, ordnete der Computer die ersten 15 Zeichen, die ich eingegeben hatte, der String-Variablen N$ zu und ignorierte den Rest einfach.

Nehmen wir an, wir lassen das Programm noch einmal laufen.

RUN
IHR NAME? JOHN JACOB JINGLEHEIMER SCHMIDT

Was wird der Computer in diesem Fall ausgeben?

....

IHR NAME IST JOHN JACOB JING

und die Lehre daraus: Dimensionieren Sie Ihre String-Variablen ausreichend!

4. Schreiben Sie ein Programm, das Autokennzeichen, bestehend aus drei Buchstaben und einer dreistelligen Zahl (z. B. SAM 123) drucken kann. Geben Sie die Buchstaben als String-Variable und die Ziffern als numerische Variable ein. Schreiben Sie auch hin, wie ein Lauf aussehen wird.

....
....
....
....
....
....
....

```
10 DIM A$(3)
20 INPUT A$,B
30 PRINT A$;B
```

```
RUN
?SAM
?123
SAM123
```

5. Sie können String-Variable auch mit Hilfe von READ- und DATA-Statements eingeben.

```
 1 REM***STRING READ/DATA KURSPLAN
 5 DIMA$(18),C$(1)
10 PRINT "KURS ","STUNDEN","GRAD"  STUFE"
20 READ A$,B$,C$
30 PRINT A$,B$,C$
40 GOTO 20
50 DATA ENGLISCH 1A,3,B,SOZ 130     ,3,A
60 DATA BUCHF. 1A,4,B,STAT.         ,3,A
70 DATA MATHEM.,3,A,GESCH.   17A,3,B,
80 DATA W.KUNDE 3A,4,C
RUN
KURS                    STUNDEN  GRAD
ENGLISCH 1A             3        B
SOZ 130                 3        A
BUCHF. 1A               4        B
STAT. 10                3        A
MATHEM.                 3        A
GESCH. 17A              3        B
W. KUNDE 3A             4        C
```

Sehen Sie sich einmal Zeile 10 des vorstehenden Programms an. Beachten Sie die im String "KURS" vorgesehenen Leerstellen. Da wir für die Bezeichnungen der einzelnen Kurse zwei Standard-Druck-Positionen reserviert haben, bewirken die zusätzlichen Leerstellen, daß "STUNDEN" an der dritten Position ausgedruckt wird und "GRAD" in der letzten Spalte. Wir hätten dies auch auf andere Weise erreichen können, beispielsweise durch einen "Null-String" an der zweiten PRINT-Position.

```
10 PRINT "KURS", "", "STUNDEN","GRAD"
```

Da jeder String mit 8 Zeichen oder weniger eine Druckspalte benötigt, haben wir auch in den DATA-Statements Leerstellen vorgesehen (siehe Zeilen 50 und 60), wo der Kursname eine Länge von 8 Zeichen oder weniger hatte. Das bewirkt, daß der Kursname zwei Druckspalten belegt. Die folgenden Regeln gelten für Strings, die in DATA-Statements enthalten sind.

1) Der erste String in einem DATA-Statement darf keine Leerstellen am Anfang haben, schließt jedoch Leerstellen bis zu dem Komma ein,

das den ersten und zweiten String voneinander trennt.

910 DATA HOW, NOW, BROWN, COW

Die Leerstellen vor NOW, BROWN und COW werden in den jeweiligen String eingeschlossen.

2) Der letzte String in einem DATA-Statement darf Leerstellen zwischen dem vorangehenden Komma und dem ersten dargestellten Zeichen des Strings enthalten, jedoch nicht hinter dem letzten Zeichen.

910 DATA HOW , NOW , BROWN , COW

Diese Leerstellen werden in die String-Zuweisung einbezogen, nicht aber Leerstellen hinter dem letzten Zeichen.

3) Die restlichen Strings in einem DATA-Statement können beliebige Leerstellen vor dem ersten oder hinter dem letzten Zeichen in der String-Zuweisung enthalten.

910 DATA HOW , NOW , BROWN , COW

Die beiden Leerstellen vor und hinter dem Wert NOW werden in die String-Zuweisung einbezogen.

6. Wie lang ist im Programm in Abschnitt 5 der String 'SOZ 130 ' in Zeile 50? . . . .

— — — — — — — — — — — — — — — — — —

9 Zeichen, einschließlich der beiden Leerstellen hinter dem letzten Zeichen.



7. Wieviele Druckspalten werden durch den String SOZ 130 belegt, wenn er aus dem DATA-Statement gelesen und durch Zeile 30 ausgedruckt wird? . . . .

— — — — — — — — — — — — — — — — — —

2; Strings mit einer Länge von 9 Zeichen oder mehr, die im PRINT-Statement durch Kommas getrennt sind, veranlassen den Computer,

die nächste PRINT-Position zu überspringen. Das erfolgt automatisch, um zu verhindern, daß auf dem Display eine Spalte in die nächste hineinreicht. Zusammenfassend kann gesagt werden, daß bei einer String-Länge über 9 Zeichen die übliche Reihenfolge der Druckspalten nicht beachtet wird, sondern die Ausgabe in der nächstmöglichen Druck-Position erfolgt.

8. Der String LET weist einer String-Variablen einen speziellen String zu. Bitte beachten Sie, daß Sie den String in Anführungszeichen setzen müssen, wie es die beiden folgenden Beispiele zeigen.

```
10 DIM A$(3),B$(2),C$(3)
20 LET A$="YES"
30 LET B$="NO"
40 LET C$=A$ <————————————— C$ enthält jetzt "YES"
```

Bitte geben Sie an, was beim Lauf dieses Programms ausgedruckt wird.

```
10 DIM A$(20), B$(20)
20 LET A$=" EIN GUTES BEISPIEL"
30 LET B$="DIES IST"
40 PRINT B$;A$
RUN
```

....

— —

DIES IST EIN GUTES BEISPIEL

9. Als nächstes möchten wir, daß Sie ein Programm schreiben, das dem in Abschnitt 5 ähnlich ist. Statt jedoch Kurse und Grade vorzusehen, sollen als Daten die Titel von Büchern verwendet werden. Jeder Datensatz enthält den Autor, den Titel und die Anzahl der Seiten. Unsere Daten sehen folgendermaßen aus.

```
900 REM***REIHENFOLGE DER DATEN
901 REM***AUTOR, TITEL, SEITEN
910 DATA BROWN,INSTANT BASIC,178
920 DATA WHITE,YOUR HOME COMPUTER,225
930 DATA KEMENY, BASIC PROGRAMMING,150
```

```
940 DATA OSBORNE,INTRO TO MICROCOMPUTERS,384
950 DATA NELSON,COMPUTER LIB/DREAM MACHINES,186
960 DATA THIAGI,GAMES FOR THE POCKET CALCULATOR,54
RUN
TITEL                  AUTOR          SEITEN
YOUR HOME COMPUTER     WHITE          225
BASIC PROGRAMMING      KEMENY         150
INTRO  TO MICROCOMP    OSBORNE        384
COMPUTER LIB/DREAM     NELSON         186
GAMES FOR THE POCK     THIAGI         54
```

Sehen Sie sich bitte, bevor Sie das Programm schreiben, die obige Zeile 940 und die entsprechende Zeile des Programmlaufs an. Wie Sie sehen, haben wir die Länge der Titel durch geeignete Dimensionierung der String-Variablen so eingeschränkt, daß sich eine einheitliche Darstellung ergibt. Das heißt: Ist ein Titel kurz, enthält also weniger als 9 Zeichen, dann sehen Sie noch leere Stellen hinter dem letzten Zeichen vor, damit er zwei Druckspalten belegt. Kürzen Sie die Länge der Namen der Autoren in der gleichen Weise.

```
  1 REM***HAUS–BIBLIOTHEK
  5 DIM A$(8),T$(18)
 10 PRINT "TITEL","","AUTOR","SEITEN"
 20 READ A$,T$,P:IF A$="END" THEN END
 30 PRINT A$,T$,P:GOTO 20
```

```
900 REM***REIHENFOLGE DER DATEN
901 REM***AUTOR, TITEL, SEITEN
910 DATA BROWN,INSTANT BASIC,178
920 DATA WHITE,YOUR HOME COMPUTER,225
930 DATA KEMENY, BASIC PROGRAMMING,150
940 DATA OSBORNE,INTRO TO MICROCOMPUTERS,384
950 DATA NELSON,COMPUTER LIB/DREAM MACHINES,186
960 DATA THIAGI,GAMES FOR THE POCKET CALCULATOR,54
970 DATA END,0,0,0
```

10. Modifizieren Sie das Programm, das Sie gerade geschrieben haben so, daß nur die Bücher mit weniger als 200 Seiten ausgedruckt werden.

. . . .

– –

```
25 IF P >= 200 THEN 20
```

11. Der String IF-THEN ermöglicht es, den Inhalt von zwei String-Variablen zu vergleichen.

```
10 DIM A$(3), B$(3)
20 B$="NO"
30 PRINT "MOECHTEN SIE INSTRUKTIONEN";
40 INPUT A$
50 IF A$=B$ THEN 140
60 PRINT "DIESE SIMULATION ERLAUBT IHNEN . . . "
140 PRINT "DIESE SIMULATION BEGINNT . . . "
```

Zeile 50 vergleicht den Inhalt der String-Variablen A$ (YES) mit dem Inhalt der String-Variablen B$ (NO). Wenn Sie auf das INPUT-Statement mit YES geantwortet haben, sind A$ und B$ nicht gleich, da sie verschiedenen Inhalt haben. Der Computer führt dann das nächste Statement in Zeile 60 aus. Hätten Sie in Zeile 40 mit NO geantwortet, verzweigt das Programm zu Zeile 140 und fährt dort mit der Ausführung fort. Der Vergleich in Zeile 50 erfolgt zwischen den . . . .

. . . .

– –

Inhalten der String-Variablen A$ und B$.

12. Sie können den Inhalt einer String-Variablen auch mit einem in Anfühtungszeichen eingeschlossenen String vergleichen.

```
10 DIM A$(4)
20 PRINT "WUENSCHEN SIE INSTRUKTIONEN? JA ODER NEIN";
30 INPUT A$
40 IF A$="NEIN" THEN 140
50 PRINT "DIESE SIMULATION ERLAUBT IHNEN . . . "
140 PRINT "DIE SIMULATION BEGINNT . . . "
```

Der Vergleich in Zeile 40 erfolgt zwischen einem . . . .
. . . . und einem . . . .

_ _

String, der einer String-Variablen zugewiesen ist (also dem Inhalt einer String-Variablen) und einem in Anführungszeichen eingeschlossenen String.

13. Eine numerische Variable dürfen Sie nicht mit einer String-Variablen vergleichen.

```
110 IF A$=B THEN 140
```

Das ist nicht erlaubt! Sie erhalten nach der Eingabe eine Fehlermeldung.

Sie können jedoch eine String-Variable, durch Verwendung der VAL($)-Funktion, in ihr numerisches Äquivalent umwandeln. Wenn im obigen Beispiel A$ eine Zahl enthält (die als String-Variable eingegeben wurde), könnten Sie sie mit der numerischen Variablen B vergleichen, indem Sie Zeile 110 folgendermaßen umschreiben:

```
110 IF VAL(A$) = B THEN 140
```

Hier sehen Sie ein weiteres Demonstrationsprogramm. Bitte kreuzen Sie an, welcher Lauf für dieses Programm richtig ist.

```
 5 DIM A$(10),B$(10),C$(10)
10 A$="32":B$="1.115":C$="-10"
30 PRINT A$:PRINT VAL(A$)
40 PRINT B$:PRINT VAL(B$)
```

```
50 PRINT C$:PRINT VAL(C$)
RUN                  RUN
32                   32
32                    32
1.115                1.115
1.115                 1.115
-10                  -10
-10                  -10
```

Der erste Lauf ist korrekt.

14. Was wird durch dieses Programm ausgedruckt?

```
 5 DIM A$(10),B$(10),C$(10)
10 A$="32":B$="1.115":C$="-10"
30 PRINT VAL(A$)+VAL(B$)+VAL(C$)
RUN
```

....

23.115

15. Mit der STR$-Funktion können Sie eine numerische Variable in einen String umwandeln, oder den Inhalt einer numerischen Variablen in eine String-Variablen einsetzen.

10 PRINT STR$(X) ◄—— Diese Anweisung veranlaßt den Ausdruck des String-Äquivalents des numerischen Inhalts der variablen X.

20 LET A$=STR$(X) ◄—— Diese Zeile weist den numerischen Inhalt der Variablen X der String-Variablen A$ zu.

Was wird durch dieses Programm ausgedruckt?

```
10 V=112 : PRINT V : PRINT STR$(V)
RUN
```

. . . .
. . . .

- -

112
112

16. In einer IF-THEN-Anweisung erfolgt der Vergleich von zwei Strings jeweils Zeichen für Zeichen. Wird beispielsweise bei einer Eingabe, an einer nicht vorgesehenen Stelle, die Leertaste betätigt, ist für den Computer keine Gleichheit mehr vorhanden.

```
 5 DIM A$(20)
10 INPUT A$
20 IF A$="MCGEE" THEN 140
```

Gibt der Benutzer, nach Aufforderung durch den Computer, jedoch MC GEE ein, wird keine Gleichheit mehr festgestellt. Warum? . . . .
. . . .

- -

Der Zwischenraum zwischen C und G repräsentiert ein Zeichen, das in "MCGEE" nicht vorhanden ist.

17. Sie können Strings mit den gleichen Symbolen vergleichen, die Sie bisher auch bereits bei Zahlen verwendet haben, also: <>, <, >, <=, >=, und =. Dabei müssen Sie jedoch sehr aufpassen. Der Vergleich erfolgt wieder zeichenweise, und zwar von links nach rechts. Der erste gefundene Unterschied bestimmt das Verhältnis beider Strings. Der Vergleich basiert auf der Position des jeweils unterschiedlichen Buchstabens im Alphabet. C ist damit "kleiner als" S, T ist "größer als" M.

```
 5 DIM A$(5),B$(5)
10 A$="SMITH"
20 B$="SMYTH"
30 IF A$<B$ THEN 100
```

Wird das obige Programm zu Zeile 100 verzweigen oder mit dem nächstfolgenden Statement fortfahren? . . . .

- -

Es wird zu Zeile 100 springen. Der erste Unterschied liegt beim dritten Zeichen vor. Da I "kleiner als" Y ist, ist die IF-THEN-Bedingung wahr.

18. Hier ist ein weiteres Beispiel

```
100 DIM D$(20),E$(20)
120 D$="COMPUTE"
130 E$="COMPUTER"
140 IF D$<E$ THEN 180
150 PRINT D$
160 END
180 PRINT E$
```

Welches Statement wird nach dem Vergleich in Zeile 140 ausgeführt?
. . . .

— — — — — — — — — — — — — — — — — — — —

Zeile 180. D$ ist "kleiner als" E$. D$ ist in diesem Fall kürzer und damit automatisch "kleiner als" E$.

19. Ändern Sie Zeile 140 in Abschnitt 18 so ab, daß sie folgendermaßen lautet.

```
140 IF D$=E$ THEN 180
```

Welches Statement wird jetzt, nach dem Vergleich von Zeile 140 ausgeführt? . . . .

— —

Zeile 150. D$ ist ungleich E$.

20. Ändern Sie Zeile 140 in Abschnitt 18 so ab, daß sie lautet:

```
140 IF E$>D$ THEN 180
```

Welches Statement wird nach dem Vergleich ausgeführt? . . . .

— —

Zeile 180. E$ ist "größer als" D$.

21. Damit Sie String-Vergleiche noch besser verstehen, möchten wir Sie jetzt mit dem ASCII-Code vertraut machen. Sicher werden Sie fragen:

"Wer oder was ist ASCII?" ASCII ist die Abkürzung von "American Standard Code for Information Interchange", also "Amerikanischer Standard-Code für den Informations-Austausch". Beispielsweise wird von der Tastatur, für jedes von Ihnen eingetippte Zeichen, der entsprechende ASCII-Code an den Computer übermittelt. Der Computer wiederum schickt für jedes Zeichen, das auf Ihrem Drucker oder Display ausgegeben wird, ein ASCII-Zeichen zurück. Das versteht man unter "Informations-Austausch", für den dieser Code geschaffen wurde.

ATARI-BASIC ermöglicht es dem Benutzer, statt der Zeichen direkt die jeweilige ASCII-Code-Nummer einzugeben. Dies geschieht mit der Funktion CHR$(X). Umgekehrt ist es möglich, ein Zeichen in seine ASCII-Code-Nummer umzuwandeln. Dazu ist die ASCII-Funktion vorgesehen.

CHR$(X) Die ASCII-Code-Nummer, die umgewandelt werden soll, wird vor der Anwendung dieser Funktion in Klammern gesetzt. Sie können aber auch, statt X, direkt die Zahl einsetzen, wie beispielsweise in CHR$(65).

ASC(X$) Das String-Zeichen, das in den ASCII-Code umgewandelt werden soll, wird vor der Ausführung für X$ eingesetzt.

Die Großbuchstaben beispielsweise, die wir in BASIC verwenden, entsprechen den Code-Nummern 65 bis 90 einschließlich: A=65, B=66, C=67, . . . X=88, Y=89, Z=90. Eine vollständige Tabelle der ASCII-Code-Nummern und der von ihnen repräsentierten Zeichen finden Sie im Anhang.

Das folgende kleine Programm druckt einen "Ausspruch" von John Wayne, der im DATA-Statement in Form von ASCII-Code-Nummern enthalten ist.

```
10 READ X : IF X=-1 THEN END
20 PRINT CHR$(X);: GOTO 10
30 DATA 89,95,80,-1
RUN
YUP
```

Bitte geben Sie an, was beim Lauf des folgenden Programms ausgedruckt wird. Verwenden Sie dazu die Tabelle der ASCII-Code-Nummern.

```
10 READ A : IF A=-1 THEN END
20 PRINT CHR$(A);:GOTO 10
30 DATA 72,65,80,89,32,67,79,77,80,85,84,73,78,71,-1
RUN
```

....

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

HAPPY COMPUTING

22. Durch Verwendung der ASC-Funktion können Sie die ASCII-Code-Nummer eines String-Zeichens ermitteln. Hier eine Demonstration.

```
 5 DIM X$(1), Y$(1),Z$(1)
10 X$="A":Y$="B":Z$="C"
20 PRINT "A= ";ASC(X$),"B= ";"ASC(Y$),"C= ";ASC(Z$)
RUN
A=65          B=66          C=67
```

Schreiben Sie ein Programm unter Verwendung der ASC-Funktion, das Ihnen die ASCII-Codes für die folgenden Zeichen ausgeibt:

```
RUN
ZEICHEN       ASCII-CODE
$             36
=             61
?             63
```

_ _

Hier sind zwei mögliche Lösungen:

```
 5 DIM A$(1),B$(1),C$(1)
10 A$="$":B$="=":C$="?"
20 PRINT "ZEICHEN","ASCII-CODE"
30 PRINT
40 PRINT A$,ASC(A$)
50 PRINT B$,ASC(B$)
60 PRINT C$,ASC(C$)
```

```
10 PRINT "ZEICHEN","ASCII–CODE"
20 PRINT
30 PRINT "$",ASC("$")
40 PRINT "=",ASC("=")
50 PRINT "?",ASC("?")
```

23. Jetzt wollen wir uns wieder mit String-Vergleichen unter Verwendung von IF-THEN-Statements beschäftigen. Strings werden, wie bereits erläutert, Zeichen für Zeichen vergleichen, wobei der Computer die ASCII-Code-Nummern zur Durchführung des Vergleichs benutzt. In Abschnitt 12 beispielsweise fanden wir folgendes Programm:

```
 5 DIM A$(10)
10 INPUT A$
20 IF A$ = "MCGEE" THEN 140
```

Wenn das Programm läuft und der Benutzer MC GEE eingibt, vergleicht der Computer die ASCII-Code-Nummern folgendermaßen:

"MCGEE"	A$ (vom Benutzer eingegeben)
M = 77	M = 77
C = 67	C = 67
G = 71	Leertaste = 32
E = 69	G = 71
E = 69	E = 69
	E = 69

Der Computer stellt fest, daß die ersten beiden Zeichen der zwei Strings gleich sind, jedoch besteht beim dritten Zeichen keine Übereinstimmung mehr (71 statt 32).

Bitte zeigen Sie, wie der Computer, unter Verwendung der ASCII-Codes, die Strings im folgenden Programm-Segment vergleicht.

```
 5 DIM A$(3)
10 INPUT "WUENSCHEN SIE INSTRUKTIONEN? JA ODER NEIN";
   A$
20 IF A$ = "NEIN" THEN 140
```

```
RUN
WUENSCHEN SIE INSTRUKTIONEN? JA ODER NEIN"JA
   A$         "NEIN"
   ....    ................
   ....    ................
   ....    ................
   ....    ................
```

Ist der Vergleich wahr oder falsch? ....

_ _

```
A$         "NEIN"
J = 74     N = 78
A = 65     E = 69
           I = 73
           N = 78
```

Der Vergleich ist falsch.

24. Bevor wir weitergehen, müssen wir noch das RESTORE-Statement und seine Verwendung in Verbindung mit READ- und DATA-Statements erläutern. Ein READ-Statement bewirkt, daß jeweils die nächste Information aus den DATA-Statements gelesen wird. Vielleicht möchten Sie aber, daß das Programm nochmals alle Daten von Anfang an liest.

Um dies zu erreichen, können Sie ein RESTORE-Statement vorsehen. Es bewirkt, das die nächste Information, die durch READ gelesen wird, wieder aus dem ersten DATA-Statement stammt. Mit anderen Worten, beim nächsten DATA-Statement, auf das der Computer trifft, beginnt er wieder am Anfang von DATA. Dies sehen Sie deutlich in Zeile 30 des folgenden Programms.

Nachdem Sie gelernt haben, wie man String-Variable miteinander vergleicht, werden Sie keine Schwierigkeiten haben, das folgende einfache Programm zu verstehen, mit dem man Informationen aus DATA-Statements herausholen kann.

Das Programm in Abschnitt 5 druckt Kurse, Stundenzahlen und den Kurs-Grad aus. Das nachfolgende Programm ermöglicht es dem Operator, den Kursnamen einzutippen, woraufhin der Computer folgende Informationen ausgibt: Kurs, Stunde, Grad.

```
  1 REM***KURS–INFORMATION
  5 DIM A$(18),D$(18),C$(1)
 10 PRINT "GEBEN SIE DEN KURSNAMEN EIN";
 20 INPUT D$
 30 RESTORE
 40 READ A$,B,C$:IF A$=D$ THEN 60
 50 GOTO 40
 60 PRINT A$ ,B,C$:PRINT : GOTO 10
900 REM***KURS, STUNDEN, GRAD
910 DATA ENGLISCH 1A,3,B,SOZ  130,3A
920 DATA BUCHF. 1A,4,B,STAT.  10,3,C
930 DATA W.KUNDE 3,A,GESCH.  17A,3,B
```

Trap (RND)

```
RUN
GEBEN SIE DEN KURSNAMEN EIN? W. KUNDE
W.KUNDE            3        A

GEBEN SIE DEN KURSNAMEN EIN? STAT 10
STAT 10            3        C

GEBEN SIE DEN KURSNAMEN EIN? ECON 1A
ERROR– 6 AT LINE 40
```

← Den Kurs gibt es nicht. Der Computer hat alle Daten durchgesucht und einen derartigen Kurs nicht gefunden. Daher druckter er eine Fehlermeldung aus.

Sehen wir uns noch einen weiteren Lauf des Programms an.

```
RUN
GEBEN SIE DEN KURSNAMEN EIN? SOZ130
ERROR– 6 AT LINE 40
```

Warum erhalten wir diesmal eine Fehlermeldung? . . . .

– – – – – – – – – – – – – – – – – – – –

Der Kursname ist als SOZ 130 gespeichert, eingegeben wurde aber SOZ130, also ohne Leerschritt zwischen SOZ und 130.

Beachten Sie bitte auch, daß Sie bei der Eingabe über die Tastatur keine Leerstellen vor oder nach der jeweiligen Zeichenfolge vorsehen können. Sie dürfen daher in Strings in DATA-Statements, die mit durch INPUT eingegebenen Strings verglichen werden sollen, keine

derartigen Leerstellen mehr vorsehen, andernfalls erfolgen die Vergleiche nicht so wie erwartet.

Bitte beantworten Sie die folgenden Fragen, die sich auf das Programm in Abschnitt 24 beziehen.

a) Welche Aufgabe hat dieser Teil von Zeile 40? ....
.... IF A$=D$ THEN 60

b) Unter welchen Bedingungen wird Zeile 50 ausgeführt? ....
50 GOTO 40

---

a) Er soll prüfen, ob es sich bei dem aus dem DATA-Statement gelesenen Kurs um den gleichen handelt, der durch das INPUT-Statement erfragt wird.
b) Wenn der durch Zeile 40 gelesene Kurs nicht der im INPUT-Statement gefragte ist.

26. Modifizieren Sie das Programm in Abschnitt 24 so, daß es statt einer Fehlermeldung die Mitteilung "DIESEN KURS GIBT ES NICHT" ausdruckt, um anzudeuten, daß der vom Benutzer angefragte Kurs nicht im Informationssystem des Computers existiert. Sie müssen zu diesem Zweck am Ende der Daten ein Flag vorsehen.

....
....

Fügen Sie dieses oder ähnliche Statements in Ihr Programm ein:

```
 45 IF A$ = "END" THEN PRINT "DIESEN KURS GIBT ES NICHT"
    : GOTO 10
940 DATA END,0,0
```

Bitte denken Sie daran: das Daten-Ende-Signal muß aus einem String, gefolgt von einem numerischen Wert, hinter dem wiederum ein String folgt, bestehen, da das READ-Statement jeweils Werte für drei Variable gleichzeitig einliest.

27. Jetzt sind Sie an der Reihe. Schreiben Sie bitte ein Programm, das die Namen und Telephonnummern Ihrer Freunde und Geschäfts-

partner enthält. Wenn Sie einen Namen eingeben, sollte der Computer mit der korrekten Telephon-Nummer antworten, wie es die folgenden Beispiele zeigen.

```
NAME? ADAM OZ
ADAM OZ            415–555–2222

NAME? JUDY WIL
JUDY WIL           112–555–0075

NAME? JERRY
ERROR– 6 AT LINE 40
```

....

....

....

....

....

....

....

....

....

....

....

....

....

....

_ _

```
  1 REM***TELEPHON–VERZEICHNIS
 10 DIM N$(8),D$(8),T$(12)
 20 PRINT "NAME";:INPUT N$
 30 RESTORE
 40 READ D$,T$
 50 IF N$=D$ THEN PRINT :PRINT D$,T$:PRINT : GOTO 20
 60 GOTO 40
900 REM***DATA:NAME, TELEPHON–NUMMER
910 DATA TONY BOD,415–555–8117
920 DATA MARY JAY,213–555–0144
930 DATA MARY MMM,213–555–1212
940 DATA JUDY WIL,112–555–0075
950 DATA ADAM OZ,415–555–2222
960 DATA BOBBY ALL,312–555–1667
```

970 DATA END,END

28. Als nächstes möchten wir Sie mit einer weiteren Funktion vertraut machen, die es Ihnen ermöglicht, Teile von Strings, "Substrings" genannt, zu manipulieren und zu untersuchen. Ein Substring ist ein Teil eines Strings und wird durch Indizes hinter der Stringvariablen gekennzeichnet, also A$(10) oder A$(1,5).

```
 5 DIM A$(30)
10 LET A$="MY HUMAN UNDERSTANDS ME"
20 PRINT A$(10)◄──────────Der Substring beginnt beim zehn-
                           ten Zeichen und schließt alle fol-
                           genden Zeichen ein.
RUN
UNDERSTANDS ME
```

Ersetzen Sie Zeile 20 durch PRINT A$(15). Was wird gedruckt, wenn die neue Zeile 20 ausgeführt wird? ....

- -

STANDS ME

29. Sehen wir uns jetzt einmal folgende Beispiele an. Um ein von Ihnen gewünschtes Zeichen aus dem String auszuwählen, geben Sie die Position dieses Zeichens innerhalb des Strings als Doppelindex an. Dadurch wird gleichzeitig das erste und letzte Zeichen des Substrings festgelegt, der in diesem Fall nur aus einem Zeichen besteht.

```
 5 DIM A$(30)
10 LET A$="MY HUMAN UNDERSTANDS ME"
20 PRINT A$(4,4)◄─────────Durch diese Anweisung wird H,
                           der vierte Buchstabe des Strings
                           ausgedruckt (ein Zwischenraum
                           zählt als ein Zeichen).
RUN
H
```

Hier sehen wir einen Substring, der beim ersten Zeichen beginnt und alle Zeichen bis einschließlich zum neunten enthält.

```
 5 DIM A$(30)
10 LET A$="MY HUMAN UNDERSTANDS ME"
20 PRINT A$(1,9)
RUN
MY HUMAN
```

Ändern Sie im obigen Programm Zeile 20 so, daß sie "PRINT A$(4,8)" lautet. Was wird jetzt bei ihrer Ausführung ausgedruckt? ....

— — — — — — — — — — — — — — — — — —

HUMAN

30. Was wird durch das folgende Programm ausgedruckt?

```
10 DIM A$(20)
20 LET A$="GAMES COMPUTERS PLAY"
30 PRINT A$(7,15),A$(17),A$(1,5)
RUN
```

....

— — — — — — — — — — — — — — — — — — — —

```
COMPUTERS        PLAY        GAMES
```

31. Hier sind Teile eines Programms, mit dem die String-Variable A$ rückwärts ausgedruckt werden kann und zwar Zeichen für Zeichen. Füllen Sie bitte die leeren Stellen aus und geben Sie an, wie der Lauf aussieht.

```
 5 DIM A$( ...................... )
10 LET A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 FOR X= ........... TO ........... STEP -1
30 PRINT A$(X, .................... ) ;
40 ...............................................
RUN
```

....

— —

```
 5 DIM A$(26)
10 LET A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 FOR X=26 TO 1 STEP -1
```

```
30 PRINT A$(X,X);
40 NEXT X
RUN
ZYXWVUTSRQPONMLKJIHGFEDCBA
```

32. Wir wollen noch einmal ein Beispiel aus dem ersten Kapitel aufgreifen, nämlich das Zahlenratespiel aus Abschnitt 9. Jetzt allerdings soll es in ein Spiel zum Erraten von Buchstaben abgeändert werden. Hier ist ein Listing des alten Programms, dessen Logik nach wie vor gilt.

```
100 REM***DIES IST EIN EINFACHES COMPUTERSPEIL
110 LET X=INT(100*RND(1))+1
120 PRINT
130 PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND
    100."
140 PRINT "ERRATEN SIE MEINE ZAHL!!!"
150 PRINT : INPUT " IHRE ZAHL"; : INPUT G
160 IF G < X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    GROESSEREN ZAHL":GOTO 150
170 IF G > X THEN PRINT "VERSUCHEN SIE ES MIT EINER
    KLEINEREN ZAHL":GOTO 150
180 IF G=X THEN PRINT"GRATULIERE!!! SIE HABEN MEINE
    ZAHL ERRATEN.":GOTO 110
```

Wir müssen jetzt zuerst eine Zeile 101 einfügen um A$ zu dimensionieren und die Variable G$ einzugeben. Ebenso muß die Funktion X$ vorgesehen werden, um den Buchstaben des Computers, der geraten werden soll, zu speichern. Zeile 105 soll ein LET-Statement mit allen Buchstaben des Alphabets enthalten, die A$ zugeordnet werden.

101 ....

105 ....

```
101 DIM A$(26),G$(1),X$(1)
105 LET A$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

33. Ändern Sie Zeile 110 jetzt so ab, daß eine Zufallszahl zwischen 1 und 26 erzeugt wird.

110 ....

———————————————————————

110 LET X=INT(26*RND(1))+1

34. Der von uns gewählte Buchstabe soll in X$ übernommen werden.

115 LET X$ ....

———————————————————————

115 LET X$ = A$(X,X)

35. Vervollständigen Sie bitte die Zeilen 140 und 150.

130 PRINT "ICH DENKE MIR EINEN BUCHSTABEN ZWISCHEN A UND Z"
140 ....
150 ....

———————————————————————

140 PRINT "ERRATEN SIE MEINEN BUCHSTABEN!!!":PRINT
150 PRINT "IHR BUCHSTABE":INPUT G$

36. Ändern Sie die Zeilen 160, 170 und 180, soweit notwendig. Lassen Sie das neue Spiel dann ablaufen.

160 ....
170 ....
180 ....

———————————————————————

Hier das gesamte Listing und ein Programmlauf:

```
100 REM***EIN EINFACHES COMPUTERSPIEL
101 DIM A$(26),G$(1),X$(1)
105 LET A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
110 LET X=INT(26*RND(1))+1
115 LET X$=A$(X,X)
120 PRINT
130 PRINT "ICH DENKE MIR EINEN BUCHSTABEN ZWISCHEN
    A UND Z"
140 PRINT "ERRATEN SIE MEINEN BUCHSTABEN!!!":PRINT
150 PRINT "IHR BUCHSTABE";INPUT G$
160 IF G$<X$ THEN PRINT "VERSUCHEN SIE ES MIT EINEM
```

```
    GROESSEREN BUCHSTABEN.":GOTO 150
170 IF G$ > X$ THEN PRINT "VERSUCHEN SIE ES MIT EINEM
    KLEINEREN BUCHSTABEN.":GOTO 150
180 IF G$ = X$ THEN PRINT "GRATULIERE!!! SIE HABEN
    MEINEN BUCHSTABEN ERRATEN.":GOTO 110

RUN
ICH DENKE MIR EINEN BUCHSTABEN ZWISCHEN A UND Z
ERRATEN SIE MEINEN BUCHSTABEN!!!
IHR BUCHSTABE L
VERSUCHEN SIE ES MIT EINEM KLEINEREN BUCHSTABEN.
IHR BUCHSTABE G
VERSUCHEN SIE ES MIT EINEM GROESSEREN BUCHSTABEN.
IHR BUCHSTABE J
GRATULIERE!!! SIE HABEN MEINEN BUCHSTABEN ERRATEN'
```

37. Schreiben Sie, unter Verwendung der Daten aus Abschnitt 27, ein Programm, das eine Liste der Telephon-Nummern derjenigen Teilnehmer ausdruckt, deren Vorname mit dem Buchstaben M beginnt.

```
RUN
MARY JAY        213–555–0144
MARY MMM        213–555--1212
ERROR– 6 AT LINE 40
```

– –

```
  1 REM***TELEPHON–VERZEICHNIS
 10 DIM D$(8), T$(12)
 20 RESTORE
 30 READ D$,T$:IF D$(1,1)="M" THEN PRINT D$,T$
 40 GOTO 30
900 REM***DATA: NAME, TELEPHON–NUMMER
910 DATA TONY BOD,415–555–8117
920 DATA MARY JAY,213–555–0144
930 DATA MARY MMM,213–555–1212
940 DATA JUDY WIL,112–555–0075
950 DATA ADAM OZ,415–555–1667
960 DATA BOBBY ALL,312–555–1667
970 DATA END,999
```

38. Modifizieren Sie Ihr Programm aus Abschnitt 37 so, daß der Computer nur die Namen und Telephon-Nummern aller Teilnehmer mit der Vorwahl 415 ausdruckt.

```
RUN
415–555–8117      TONY BOD
415–555–2222  ADAM OZ
ERROR– 6 AT LINE 40
```

– –

```
  1 REM***TELEPHON–VERZEICHNIS
 10 DIM D$(8),T$(12)
 20 RESTORE
 30 READ D$,T$:IF T$(1,3) <> "415" THEN 30
 40 PRINT T$,D$:GOTO 30
900 REM***DATA:NAME, TELEPHON–NUMMER
910 DATA TONY BOD,415–555–8117
920 DATA MARY JAY,213–555–0144
930 DATA MARY MMM,213–555–1212
940 DATA JUDY WIL,112–555–0075
950 DATA ADAM OZ,415–555–1667
970 DATA END,999
```

39. Manchmal ist es notwendig, die Länge des Inhalts einer String-Variablen zu kennen (Anzahl der Zeichen). Natürlich ist in BASIC auch dafür eine spezielle Funktion vorgesehen, nämlich LEN. LEN(A$) gibt Ihnen die Anzahl der Zeichen, einschließlich der Leerstellen, im String A$ an. Hier sind einige Beispiele.

```
10 LET A = LEN(A$)
20 PRINT LEN(C$)
30 IF LEN(A$) <> LEN(B$) . . .
```

Sie können Substrings dazu benutzen, um die Antworten auf Fragen des Computers zu überprüfen, und entsprechende Reaktionen zu veranlassen, usw. Nachfolgend finden Sie einen Teil eines Geschichts-Lehrprogramms. Das Programm stellt eine Frage und fordert den Benutzer zu einer Antwort auf. Anschließend überprüft es die Antwort und macht eine entsprechende Bemerkung.

```
10 DIM A$(30)
20 PRINT "WER WAR DER ERSTE PRAESIDENT DER USA";:
   INPUT A$
30 FOR X=1 TO LEN(A$)-9
40 IF A$(X,9+X)="WASHINGTON" THEN PRINT "RICHTIG":END
50 NEXT X
60 PRINT "IHRE ANTWORT IST FALSCH"
70 GOTO 20
```

```
RUN
WER WAR DER ERSTE PRAESIDENT DER USA
```

Was wird der Computer antworten, wenn Sie GEORGE WASHINGTON eingeben? ....

_ _

RICHTIG. Um zu sehen, wie der Computer Ihre Antwort überprüft, fügen Sie die folgende Zeile ein und lassen Sie das Programm noch einmal mit der gleichen Antwort laufen.

```
39 PRINT A$(X,X+9)
```

40. Was antwortet der Computer, wenn der Benutzer, als Antwort auf die Frage des vorigen Programms, WASHINGTON IRVING eingibt?

....

_ _

RICHTIG. Wir haben die Antwort nämlich auf WASHINGTON überprüft.

41. Die LEN-Funktion kann auch dazu verwendet werden, zu veranlassen, daß vom Anwender eingegebene Daten eine bestimmte Länge erhalten, wenn Sie ein vorgegebenes Format nicht überschreiten dürfen, beispielsweise weil sie in ein Formular eingesetzt werden sollen. Sehen Sie sich einmal das nachfolgende Beispiel an und beantworten Sie dann die Fragen.

```
10 DIM A$(30),N$(30)
20 PRINT "IHR NAME";:INPUT N$
30 IF LEN(N$) > 20 THEN PRINT "BEGRENZEN SIE IHREN
   NAMEN BITTE AUF 20 ZEICHEN";GOTO 20
```

```
40 PRINT "IHRE ADRESSE";:INPUT A$
50 IF LEN (A$) > 15 THEN PRINT "BITTE KUERZEN SIE IHRE
   ADRESSE AB":GOTO 30
```

Wieviele Zeichen sind für den Namen erlaubt? . . . .
Wieviele für die Adresse? . . . .

_ _

20; 15

EINGENTEST

1. Schreiben Sie bitte ein Programm, das die Eingabe eines Wortes mit fünf Buchstaben ermöglicht und dieses Wort dann rückwärts ausdruckt.

   . . . .
   . . . .
   . . . .
   . . . .
   . . . .
   . . . .
   . . . .

2. Schreiben Sie ein Programm, durch das eine Reihe von Wörtern mit einer Länge von vier Buchstaben aus DATA-Statements eingelesen wird und drucken Sie nur die Worte, die mit dem Buchstaben A beginnen.

   . . . .
   . . . .
   . . . .
   . . . .
   . . . .
   . . . .

3. Modifizieren Sie das Programm in Frage 2 so, daß nur die Worte gelesen werden, die mit A beginnen und mit S enden.

   . . . .
   . . . .

4. Vor einigen Jahren hatte die amerikanische Auto-Industrie erhebliche Probleme, Namen für neue Autos zu erfinden. Man setzte daher einen Computer ein, um Worte mit fünf Buchstaben zu erzeugen. Schreiben Sie ein Programm, durch das 100 Worte mit je fünf Buchstaben ausgegeben, bei denen für die erste, dritte und fünfte Stelle zufällig ausgewählte Konsonanten und für die zweite und vierte Position auf die gleiche Weise zufällig ermittelte Vokale eingesetzt werden.

1.
```
10 REM***STRING-EIGENTEST 9-1
20 DIM A$(5)
30 INPUT A$
40 FOR X=5 TO 1 STEP -1
50 PRINT A$(X,X);
60 NEXT X
70 PRINT :GOTO 30
```

2.
```
10 REM***STRING-EIGENTEST 9-2
20 DIM A$(4)
30 READ A$:IF A$(1,1) <> "A" THEN 30
40 PRINT A$:GOTO 30
50 DATA ANTS,GNAT,LOVE,BALD,APES
60 DATA BAKE,MIKE,KARL,BARD,ALAS
```

3.
```
10 REM***STRING-EIGENTEST 9-3
20 DIM A$(4)
30 READ A$:IF A$(1,1) <> "A" THEN 30
35 IF A$(4,4) <> "S" THEN 30
40 PRINT A$:GOTO 30
50 DATA ANTS,GNAT,LOVE,BALD,APES
60 DATA BAKE,MIKE,KARL,BARD,ALAS
```

4.
```
10 REM***STRING-EIGENTEST 9-4
20 DIM A$(5),B$(21)
30 A$="AEIOU"
40 B$="BCDFGHJKLMNPQRSTVWXYZ"
50 FOR X=1 TO 20
60 FOR Z=1 TO 2
70 B=INT(21*RND(1))+1
80 PRINT B$(B,B);
90 A=INT(5*RND(1))+1
100 PRINT A$(A,A),
110 NEXT Z
120 B=INT(21*RND(1))+1
130 PRINT B$(B,B)
140 NEXT X
```

NOTIZEN

KAPITEL 10

Farb-Graphik und Ton

In diesem Kapitel werden wir Ihnen einige der interessantesten Möglichkeiten des ATARI-Computers vorstellen. Sie werden lernen, die Farbe der Objekte, die auf dem Bildschirm erscheinen, zu beeinflussen und ebenso die Farbe des jeweiligen Hintergrundes. Darüber hinaus werden Sie erfahren, wie man Bilder auf dem Display "malt", die Sie selbst entworfen haben. Diese zusätzlichen Möglichkeiten für die Verbesserung der Ausgabe Ihrer Programme werden durch eine Reihe neuer Graphik- und Farb-Statements und Befehle gegeben. Wenn Sie dieses Kapitel beendet haben, können Sie .....

- für den Hintergrund Ihres Displays eine von 16 Farben auswählen;
- Entwürfe auf dem Bildschirm anfertigen, unter Verwendung einer von 16 möglichen Farben;
- einen Text zu Ihren Bildern in einem speziellen "Text"-Fenster ausdrucken;
- unter 16 Farben die von Ihnen gewünschte für das Textfenster auswählen;
- die Nummer der jeweiligen Zeichen-Positionen ändern, so daß detailliertere Bilder gezeichnet werden können.

1. Durch Auswahl einer Bildschirm-Betriebsart, welche die Mischung von Graphik und Text auf verschiedene Arten ermöglicht, können Sie Linien auf dem Bildschirm ziehen. Dazu muß das Kommando GRAPHICS gegeben werden, das sich mit GR. abkürzen läßt. Acht verschiedene Betriebsarten sind vorgesehen. Sie werden durch die Zahlen 0 bis 7 ausgewählt. Beispielsweise würde

```
10 GR. 3
```

ein graphisches Raster mit 40 Spalten und 20 Reihen auswählen, bei dem 4 Zeilen für Text vorgesehen sind.

a) Welches Kommando ermöglicht das Zeichnen von Graphiken? .....

b) Wie lautet seine Abkürzung? ....

c) Wieviele verschiedene Bildschirm-Betriebsarten gibt es? ....

— — — — — — — — — — — — — — — — — — —

a) GRAPHICS b) GR. (Der Punkt ist wichtig!) c) 8 (0 . . . 7)

2. Tippen Sie folgende Zeile ein und beobachten Sie den Bildschirm.

GRAPHICS 3

a) Hat sich die Farbe im oberen Bildschirmteil geändert? ....

b) Welche Farbe hat der untere Teil des Bildschirms? ....

— —

a) Der obere Teil des Bildschirms ist jetzt leer

b) Purpur (kann abweichend sein, je nach Fernsehgerät)

3. In der Graphik-Betriebsart 3 (Graphik-Mode 3) enthält der obere Teil des Bildschirms ein Graphikraster mit 40 x 20 Positionen zum Zeichnen von Punkten. Der kleinere Teil des Bildschirms, am unteren Rand, ist für die Ausgabe von Text vorgesehen. Hier können vier Zeilen beschrieben werden.

a) Wo können Sie im Graphik-Mode 3 Punkte auf dem Bildschirm zeichenen? ....

b) In welchem Bereich kann Text ausgegeben werden? ....

— —

a) Im oberen Bildschirmbereich. b) Im unteren Bereich

4. Bis jetzt haben wir im gesamten Buch die Graphik-Betriebsart 0 (Graphics 0) benutzt. Wenn Sie Ihren ATARI-Computer einschalten, wird der Bildschirm automatisch in diese Betriebsart versetzt. Bitte denken Sie bei der Beantwortung der folgenden Frage daran, daß GRAPHICS 3 vier Zeilen Text unterhalb der Graphik-Fläche ermöglicht. Geben Sie das folgende Programm ein und lassen Sie es ablaufen. Wenn Sie es beendet haben, tippen Sie GRAPHICS 0 ein und drücken die Taste RETURN. Damit sind Sie wieder in der Text-Betriebsart (Text-Mode).

```
10 GRAPHICS 3
20 PRINT "LINE 1"
30 PRINT "LINE 2"
40 PRINT "LINE 3"
50 PRINT "LINE 4";
60 GOTO 60
```

Was können Sie jetzt am unteren Rand des Bildschirms lesen?

. . . .

. . . .

. . . .

. . . .

– –

LINE 1
LINE 2
LINE 3
LINE 4

5. Hier sehen Sie eine Tabelle der Graphik-Betriebsarten 3 bis 7.

Betriebs-Art Nr.	Anzahl der Spalten	Graphik-Positionen Reihen	Text-Fenster
3	39	20	4 Zeilen
3,5	79	40	4 Zeilen
6,7	158	80	4 Zeilen

Welche Graphik-Betriebsarten bieten die größte Anzahl graphischer Positionen? . . . .

– –

6 und 7 (158 x 80 Positionen)

6. Ändern Sie Zeile 10 aus dem Programm in Abschnitt 4 in:

```
10 GRAPHICS 4
```

Haben Sie mehr Platz für Text, wenn dieses geänderte Programm abläuft? . . . .

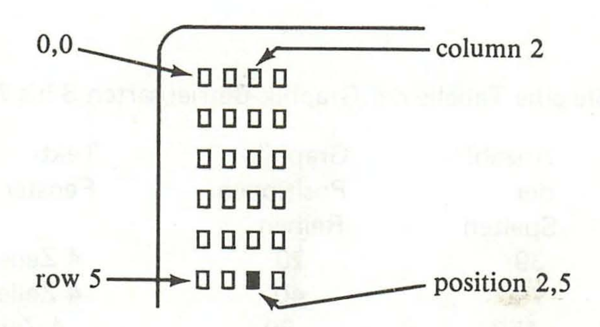
– –

Nein, es sind immer noch vier Zeilen.

7. Wenn wir graphische Darstellungen anfertigen möchten, verwenden wir das PLOT-Statement, um dem Computer mitzuteilen, welchen Punkt er darstellen soll. In diesem Statement müssen sowohl die gewünschte Spalte als auch die zugehörige Zeile angegeben werden. Die Bildschirm-Positionen werden von der linken oberen Ecke des Bildschirms aus numeriert, jeweils beginnend mit der Nummer 0. Die Position 2,5 liegt daher in der dritten Spalte nach rechts und in der sechsten Zeile nach unten. Nehmen wir einmal an, wir führen das Statement

30 PLOT 2,5 aus.

Dieser Punkt wird auf dem Bildschirm folgendermaßen lokalisiert:



Welchen Auftrag gibt das PLOT-Statement dem Computer? ....
....

— — — — — — — — — — — — — — — — — — —

Einen Punkt an einer vorgegebenen Position festzulegen (Reihe, Spalte).

8. Sehen Sie sich dieses Programm an:

```
10 GR.3
20 PLOT 2,5
30 PRINT "DAS IST DIE PLOT–POSITION 2,5"
```

Welche Bedeutung hat jede einzelne Zeile dieses Programms?

a) Zeile 10 wählt ....
b) Zeile 20 gibt an, wo ....
c) Zeile 30 ....

– –

a) Zeile 10 wählt die Graphik-Betriebsart 3
b) Zeile 20 sagt, wo der Punkt liegen soll.
c) Zeile 30 gibt an, welcher Text in das Text-Fenster geschrieben werden soll.

9. Um den Punkt sehen zu können, wählen wir mit folgendem Statement eine Farbe aus:

15 COLOR 1

Fügen Sie diese Zeile in das Programm in Abschnitt 8 ein und lassen Sie das Programm laufen.

a) Sehen Sie den Punkt? ....
b) Wenn Zeile 20 in PLOT 10,5 abgeändert wird, liegt der Punkt dann rechts oder unterhalb des ursprünglichen Punktes? ....

– –

a) Ja, der Punkt ist gelb (goldfarben) b) rechts

10. Um von GR.3 den Text-Betrieb zurückzukehren, tippen Sie einfach GR.0 ein. Schreiben Sie dann ein Programm, mit dem die nachfolgend angegebenen vier Punkte abgebildet werden. Verwenden Sie GRAPHICS 3 und COLOR 1.

1) 5,5
2) 15,5
3) 5,15
4) 15,15

....
....
....
....

– –

Unser Programm:
10 GR.3
20 COLOR 1

```
30 PLOT 5,5:PLOT 15,5
40 PLOT 5,15: PLOT 15,15
```

11. Von welchen Punkten (1,2,3 und 4) werden die angegebenen Positionen belegt, wenn das Programm läuft?

a) oben links ....
b) unten rechts ....
c) oben rechts ....
d) unten links ....

– –

a) 1 (5,5) b) 4 (15,15) c) 2 (15,5) d) 3 (5,15)

12. Vervollständigen Sie das folgende Programm, um im Graphik-Mode 3 in jeder Ecke der Graphik-Fläche einen Punkt darzustellen. Denken Sie dabei daran, daß Reihen und Spalten von 0 ab gezählt werden. Sehen Sie sich auch die Tabelle in Abschnitt 5 an. Tippen Sie zunächst GR.0 ein, um den ganzen Bildschirm für Text zur Verfügung zu haben. Geben Sie dann Ihr fertiges Programm ein und probieren Sie es aus.

```
10 GR.3
20 COLOR 1
30 PLOT 0,0: PLOT ..............................
40 PLOT 0,19: PLOT .............................
```

– –

```
30 PLOT 0,0: PLOT 38,19
40 PLOT 0,19: PLOT 38,0
```

13. Gemäß der Tabelle in Abschnitt 5 stehen uns in den Graphik-Betriebsarten 4 und 5 79x40 Punkte zur Verfügung. Da die Spalten und Reihen von 0 an numeriert sind, wird die obere rechte Ecke des Schirms durch die folgenden Statements markiert:

```
10 GR.4
20 COLOR 1
30 PLOT ........................................
```

– –

```
30 PLOT 78,0
```

14. Schreiben Sie im Graphik-Mode 6 ein Programm, das für diese Betriebsart in jeder Ecke des Bildschirms einen Punkt abbildet. Sehen Sie sich noch einmal Abschnitt 12 an, in dem diese vier Punkte für die Betriebsart 3 programmiert wurden.

....

....

....

....

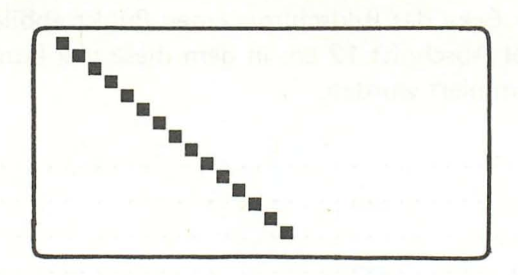
— —

Unser Muster-Programm:

```
10 GR.6
20 COLOR 1
30 PLOT 0,0: PLOT 157,79
40 PLOT 0,79: PLOT 157,0
```

15. Geben Sie wieder GRAPHICS 0 ein, um in die Text-Betriebsart zurückzukehren. Tippen Sie anschließend NEW ein und lassen Sie das folgende Programm laufen. Machen Sie dann in das freie Feld eine Skizze der Darstellung auf dem Bildschirm. Kehren Sie danach wieder in die Text-Betirebsart zurück.

```
10 GRAPHICS 3
20 COLOR 1
30 PLOT 0,0: PLOT 1,1: PLOT 2,2
40 PLOT 3,3: PLOT 4,4: PLOT 5,5
50 PLOT 6,6: PLOT 7,7: PLOT 8,8
60 PLOT 9,9: PLOT 10,10: PLOT 11,11
70 PLOT 12,12: PLOT 13,13: PLOT 14,14
80 PLOT 15,15: PLOT 16,16: PLOT 17,17
90 PLOT 18,18: PLOT 19,19
```



16. Sie können auch eine FOR-NEXT-Schleife verwenden, um eine Folge von Punkten zu zeichnen. Geben Sie dazu das folgende Programm ein und lassen Sie es ablaufen.

```
10 GRAPHICS 3
20 COLOR 1
30 FOR I=0 TO 19 STEP 2
40 PLOT I,I
50 NEXT I
```

Wie sieht das Ergebnis dieses Laufs im Vergleich zu dem von Programm 14 aus? ....

Man erhält die gleiche Darstellung.

17. Eine dritte Methode zum Zeichnen einer geraden Linie besteht darin, zusammen mit dem PLOT-Statement das DRAWTO-Statement zu verwenden. Das PLOT-Statement sagt dem Computer, wo er mit der Linie beginnen soll, während das DRAWTO-Statement angibt, wo die Linie enden soll. Die Linie wird zwischen der PLOT- und der DRAWTO-Position gezogen.

Geben Sie das folgende Programm ein und lassen Sie es ablaufen.

```
10 GR.3
20 COLOR 1
30 PLOT 0,0
40 DRAWTO 19,19
```

Vervollständigen Sie diese Statements.

a) Das PLOT-Statement weist den Computer an, ...
b) Das DRAWTO-Statement gibt dem Computer an, ...
c) Vergleichen Sie das Ergebnis dieses Programms mit der Ausgabe der beiden vorigen Programme zum Zeichnen von Linien (Abschnitte 15 und 16). ....

---

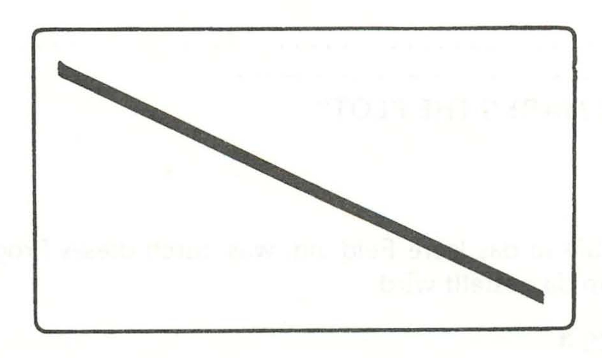
a) die Linie an der gegebenen Position zu beginnen.
b) an welcher Position er die Linie beenden soll.
c) gleiches Ergebnis.

18. Vervollständigen Sie dieses Programm, durch das eine Linie von der linken oberen in die rechte untere Ecke des Bildschirms gezogen wird. Verwenden Sie die Graphik-Betriebsart 4.

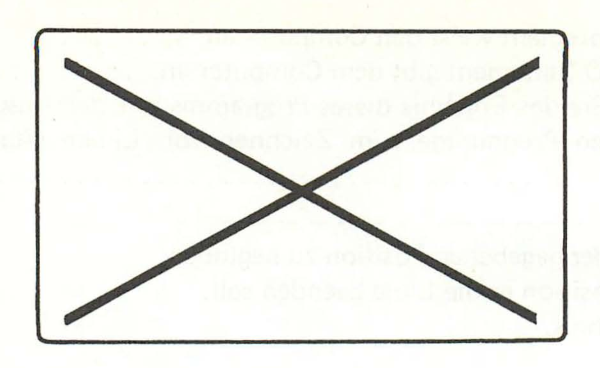
```
10 GR. 4
20 COLOR 1
30 PLOT 0,0
40 ...............................................
```

```
40 DRAWTO 78,39
```

19. Wenn das Programm in Abschnitt 18 abläuft, sollte folgendes auf dem Bildschirm sichtbar sein:



Fügen Sie jetzt zwei weitere Zeilen in das Programm aus Abschnitt 18 ein, so daß eine zweite Linie von der rechten oberen zur linken unteren Ecke gezogen wird. Auf Ihrem Display sollte jetzt ein großes X sichtbar sein, so wie es die folgende Abbildung zeigt:



Vervollständigen Sie diese Zeilen und fügen Sie sie in das Programm aus Abschnitt 18 ein.

50 PLOT ....
60 DRAWTO ....

----

50 PLOT 78,0
60 DRAWTO 0,39

20. Fügen Sie jetzt noch eine weitere Zeile in das gleiche Programm ein, so daß im Text-Fenster steht: X MARKS THE PLOT

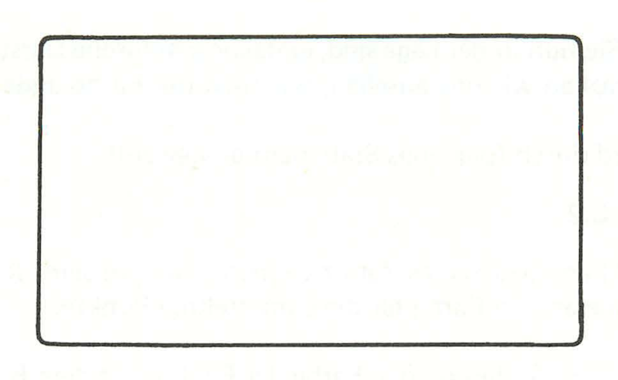
70 ....

----

70 PRINT "X MARKS THE PLOT"

21. Zeichnen Sie in das leere Feld ein, was durch dieses Programm auf dem Bildschirm dargestellt wird.

```
10 GRAPHICS 4
20 COLOR 1
30 FOR N=39 TO 0 STEP -13
40 PLOT 0,0
50 DRAWTO 78,N
60 NEXT N
```



---

22. Fügen Sie die folgenden Zeilen in das Programm aus Abschnitt 21 ein. Nachdem das Programm gelaufen ist betätigen Sie die Taste BREAK, um es anzuhalten. Geben Sie dann GRAPHICS 0 ein, um wieder in die Text-Betirebsart zurückzukehren.

Einfügen:	15 FOR M = 0 TO 1
Ändern:	20 COLOR M
Einfügen:	70 NEXT M
Einfügen:	80 GOTO 15

Wie wirken sich diese Änderungen auf das Ergebnis aus? . . . .

. . . .

Sie können sehen, wie jede Linie nacheinander gezeichnet und wieder "gelöscht" wird.

23. Nachdem Sie nun in der Lage sind, einfache graphische Darstellungen zu erzeugen, wollen wir uns ansehen, wie man die Farbe ändern kann.

Eine Farbe wird durch folgendes Statement ausgewählt:

SETCOLOR 0,C,2

Die Variable C kann jede ganze Zahl zwischen 0 und 15 sein. Jeder Wert von C ergibt eine andere Farbe für die dargestellten Punkte.

Das Statement zur Änderung der Farbe der Punkte auf dem Bildschirm lautet: ....

— —

SETCOLOR 0,C,2

24. Ein vollständiges Farb-Graphik-Programm muß die drei folgenden Statements enthalten: GRAPHICS, COLOR und SETCOLOR. Natürlich müssen wir auch noch einige Punkte definieren.

```
10 GR.3
20 COLOR 1
30 SETCOLOR 0,0,2
40 PLOT 0,3
50 DRAWTO 38,3
60 PRINT "WELCHE FARBE SEHEN SIE?"
```

Sehen Sie sich das Programm genau an, bevor Sie es ablaufen lassen. Was wird Ihrer Meinung nach auf dem Bildschirm zu sehen sein?

a) Im Graphik-Bereich? ....
b) Im Text-Bereich? ....

— —

a) ein langer, grauer Balken (die Farbe kann bei Ihrem Fernsehgerät abweichend sein).
b) Die Worte: WELCHE FARBE SEHEN SIE?

25. Mit einigen Änderungen und Einfügungen kann das Programm aus Abschnitt 24 so abgwandelt werden, daß jede der 16 verfügbaren

Farben sichtbar werden.

a) Vervollständigen Sie das Programm:

```
              10 GR. 3
              20 COLOR 1
Einfügen:     25 FOR N=0 TO 15
Ändern:       30 SETCOLOR 0, .............
              40 PLOT 0,3
              50 DRAWTO 38,3
Ändern:       60 FOR W=1 TO 500: NEXT.................
Einfügen:     70 PRINT : PRINT : PRINT : PRINT N
Einfügen:     80 NEXT ...........................
```

b) Beschreiben Sie bitte, was beim Lauf des Programms geschieht:

....

....

– –

a) 30 SETCOLOR 0,N,2
   60 FOR W=1 TO 500: NEXT W
   80 NEXT N

b) Der Farbbalken varriert von grau über alle verfügbaren Farben bis hin zu Grün, wobei jede Farbe in mehreren Helligkeitswerten sichtbar wird.

26. Sie haben gerade die 16 Farben gesehen, die bei Ihrem ATARI-Computer zur Verfügung stehen. Sie wurden durch Änderung der Farbe des Balkens im Graphik-Feld demonstriert. Jetzt wollen wir noch eine weitere Variante ausprobieren. Nehmen Sie dazu folgende Änderung in Zeile 30 des Programms aus Abschnitt 25 vor:

30 SETCOLOR 2,N,2

Lassen Sie dieses geänderte Programm ablaufen und beschreiben Sie, was geschieht. ....

....

– –

Der Farbbalken bleibt während des ganzen Laufs goldfarben, aber das Text-Fenster nimmt nacheinander jeder der 16 möglichen Farben an.

27. Eine weitere Variation ist durch eine nochmalige Änderung von Zeile 30 möglich.

Ändern Sie Zeile 30 in: 30 SETCOLOR 4,N,2

Lassen Sie das Programm auch mit dieser Änderung ablaufen. Beschreiben Sie, was geschieht. ....

....

– – – – – – – – – – – – – – – – – – – –

Diesmal ändert sich der Hintergrund der Graphikfläche. Der Balken bleibt golden, während das Text-Fenster dunkel wird und während des ganzen Laufs auch dunkel bleibt.

28. In den Abschnitten 25, 26 und 27 haben wir die Werte der Variablen A und N im Statement SETCOLOR A,N,2 verändert.

a) Wo änderten sich auf dem Bildschirm die Farben, als A auf 0 gesetzt wurde? ....

b) Wo änderten sich die Farben, als A auf 2 gesetzt wurde? ....

c) Wo änderten sich die Farben, als A auf 4 gesetzt wurde? ....

– –

a) im abgebildeten Balken

b) im Text-Fenster

c) im Hintergrund des Graphik-Bereichs.

29. Die in den Abschnitten 25, 26 und 27 demonstrierten Effekte können auch in einem Programm vereint werden.

a) Füllen Sie dazu die leeren Stellen in den Zeilen 60, 80, 90 und 100 aus:

```
10 GR.3
20 COLOR 1
30 PRINT "BEOBACHTEN SIE, WIE SICH MEINE FARBEN
   AENDERN!!!"
40 FOR M = 0 TO 4 STEP 2
50 FOR N = 0 TO 15
60 SETCOLOR ...................................
70 PLOT 5,5: DRAWTO 38,5
80 FOR W = 1 TO 200:NEXT ......................
90 NEXT .......................................
```

100 NEXT ....

b) Beschreiben Sie kurz das Resultat dieser Änderungen:

....

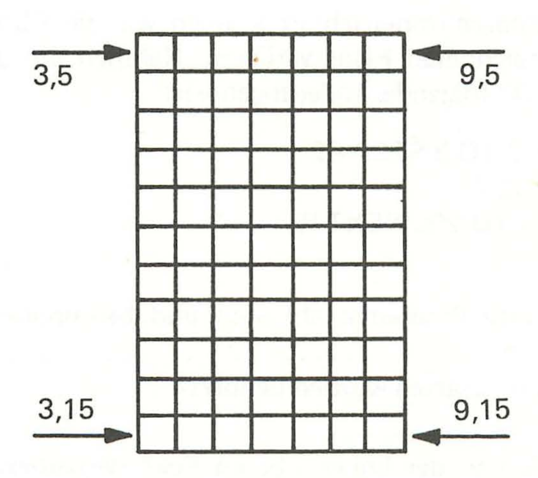
....

---

a) 60 SETCOLOR M,N,2
80 FOR W = 1 TO 200: NEXT W
90 NEXT N
100 NEXT M

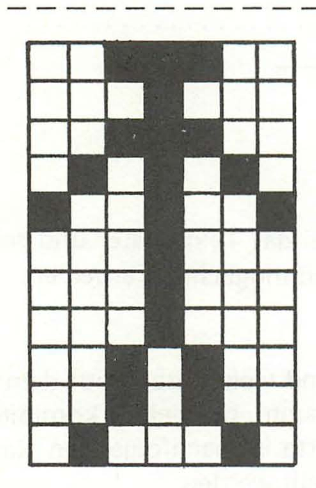
b) Zuerst nimmt der Farbbalken, dann das Textfenster und schließlich der Hintergrund nacheinander alle 16 möglichen Farben an.

30. Sie können jetzt Punkte abbilden und viele Farben auf dem Bildschirm sichtbar machen. Hier ist nun ein Programm, das beides komibiniert. Bevor Sie es laufen lassen, schraffieren Sie bitte im nachfolgenden Rasterfeld die Punkte, die durch das Programm dargestellt werden.




```
10 GR.3
20 COLOR 1
30 SETCOLOR 4,1,2
40 PLOT 5,5: DRAWTO 7,5
50 PLOT 6,6: DRAWTO 6,10
60 DRAWTO 4,15
70 PLOT 6,10: DRAWTO 8,15
```

```
80 PLOT 5,7: DRAWTO 3,9
90 PLOT 7,7: DRAWTO 9,9
100 PRINT "WAS FUER EIN LUSTIGER MANN!!!"
```



31. Durch Ändern der Bildschirmbetriebsart können wir die Abmessungen der in Abschnitt 30 gezeichneten Figur variieren. Nehmen Sie dazu beim Programm aus Abschnitt 30 folgende Änderungen vor:

Einfügen:	5 FOR A = 7 TO 3 STEP –2
Ändern:	10 GRAPHICS A
Einfügen:	93 FOR W=1 TO 200 NEXT W
Einfügen:	95 NEXT A

a) Lassen Sie das geänderte Programm ablaufen und beschreiben Sie das Ergebnis. . . . .

b) Wieviele Bildschirm-Betriebsarten wurden benutzt? . . . .

----

a) Der Mann scheint sich von der linken oberen Ecke wegzubewegen und wird dabei immer größer.

b) 3

32. Das Programm aus Abschnitt 31 änderte die Graphik-Betriebsart von vielen kleinen Feldern zu wenigen großen. Bilder können jedoch wesentlich detailreicher dargestellt werden, wenn viele kleine statt weniger großer Punkte verwendet werden.

Die folgenden Fragen beziehen sich auf das Programm aus Abschnitt 31.

a) Welches Statement bewirkte die Änderung der Graphik-Betriebsart? ....

b) Der erste Wert für die Variable A der FOR-NEXT-Schleife lautete: ...., der zweite .... und der dritte ....

---

a) Zeile 10 GRAPHICS A   b) 7, 5, 3

33. Die Tabelle in Abschnitt 5 gibt die Anzahl der Graphik-Positionen in den verschiedenen Betriebsarten an. Die Betriebsarten 6 und 7 haben 158 x 80 oder 12640 individuelle Punkte, die Betriebsart 4 und 5 79x40 oder 3160 Punkte. Betriebsart 3 verfügt über nur 39x20 = 780 Punkte. Dieses Programm gibt Ihnen nun die Möglichkeit, einige Rechtecke in verschiedenen Größen zu untersuchen.

```
10 GRAPHICS 7
20 COLOR 1
30 SETCOLOR 0,15,2
35 X=80:Y=48
40 PLOT X,Y
50 PLOT X–2,Y–2:DRAWTO X+2,Y–2
60 DRAWTO X+2,Y+2:DRAWTO X–2,Y+2
70 DRAWTO X–2,Y–2
80 PLOT X–4,Y–4:DRAWTO X+4,Y–4
90 DRAWTO X+4,Y+4:DRAWTO X–4,Y+4
100 DRAWTO X–4,Y–4
```

Lassen Sie zunächst das Programm ablaufen und beobachten Sie die Breite der gezogenen Linien sowie die Größe des Rechtecks.

Ändern Sie dann Zeile 10 in   10 GRAPHICS 5   und
Zeile 35 in   35 X=40: Y=24   um.

Lassen Sie das gleiche Programm noch einmal laufen und vergleichen Sie die Linienbreite und die Größe des Rechtecks mit dem ersten Lauf.

Ändern Sie jetzt nochmals Zeile 10 in   10 GRAPHICS 3
und Zeile 35 in   35 X=20:Y=12   um.

Lassen Sie das Programm wiederum ablaufen.

a) Wie änderte sich die Linienbreite für Lauf 1,2 und 3? ....

b) Wie änderte sich die Größe des Rechtecks? ....

_ _

a) Sie wurde jedesmal dicker. b) es wurde jeweils größer.

34. Sie können einen interessanten optischen Effekt erzielen, wenn Sie alle drei Graphik-Betriebsarten in einem Programm vereinen. Unser geändertes Programm sieht folgendermaßen aus.

```
10 M=1
20 FOR N=7 TO 3 STEP -2
30 GRAPHICS N
40 COLOR 1
50 SETCOLOR 0,1,2
60 X=80/M: Y=48/M
70 PLOT X,Y
80 PLOT X-2,Y-2: DRAWTO X+2,Y-2
90 DRAWTO X+2,Y+2: DRAWTO X-2,Y+2
100 DRAWTO X-2,Y-2
110 PLOT X-4,Y-4: DRAWTO X+4,Y-4
120 DRAWTO X+4,Y+4: DRAWTO X-4,Y+4
130 DRAWTO X-4,Y-4
140 FOR W=1 TO 200: NEXT W
150 M=2*M
160 NEXT N
170 GOTO 10
```

Bevor Sie das Programm ablaufen lassen, beantworten Sie bitte folgende Fragen:

a) Welche Zeile ermöglicht es ihnen, die Farbe des Rechtecks zu ändern? ....

b) Welche Taste würden Sie betätigen, um das Programm zu stoppen? ....

c) Welche Zeile ermöglicht es Ihnen, die Zeitdauer zu ändern, für die jede Graphik-Betriebsart auf dem Bildschirm bleibt? ....

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

a) Zeile 50 b) BREAK c) Zeile 140

35. Da Sie die Erzeugung von Farb-Graphiken jetzt wohl einigermaßen im Griff haben dürften, wollen wir uns als nächstes mit der Programmierung von Tönen befassen. Vier Variable sind es, die in Zusammenhang mit dem SOUND-Statement benutzt werden. Wir nennen sie V (für Voice), N (für Note), T (für Ton) und L (für Lautstärke).

```
20 SOUND V,N,T,L
```

Bitte geben Sie an, welche Bedeutung jede Variable im obigen Statement hat.

a) V = ...
b) N = ...
c) T = ...
d) L = ...

— — — — — — — — — — — — — — — — — — — —

a) V = Voice (Stimme) b) N = Note c) T = Ton d) L = Lautstärke

36. Geben Sie das folgende Programm ein und benutzen Sie es, um die Lautstärke Ihres Fernsehgerätes auf einen annehmbaren Wert einzustellen. Drücken Sie auf BREAK, wenn Sie damit fertig sind.

```
10 SOUND 0,115,10,8
20 GOTO 20
```

Bitte geben Sie an, welche Variable jeder der vier im SOUND-Statement enthaltenen Werte repräsentiert.

a) 8 ....
b) 115 ....
c) 0 ....
d) 10 ....

— —

a) 8 = Lautstärke b) 115 = Note c) 0 = Voice d) 10 = Ton

37. Zuerst wollen wir mit der Lautstärke experimentieren. Der Wert für L kann jede beliebige ganze Zahl zwischen 0 und 15 sein.

Verwenden Sie 0 für Voice, 115 für Note und 10 für Ton und schreiben Sie ein Programm, durch das alle 16 Lautstärkewerte erzeugt werden.

Sehen Sie für L ein INPUT-Statement vor und fügen Sie das Statement

IF L = –5 THEN ....

ein, um den Ton abzuschalten, sobald es gewünscht wird.

....
....
....
....
....

— —

Hier unser Programm:

```
10 INPUT L
20 IF L=–5 THEN 50
30 SOUND 0,115,10,L
40 GOTO 10
50 END
```

38. Im Programm aus Abschnitt 37 wird jede Lautstärke solange aufrechterhalten, bis ein neuer Wert eingegeben wird. Schreiben Sie daher bitte jetzt ein neues Programm, das eine Zeitverzögerung verwendet, um den Ton für eine vorgegebene Zeit erklingen zu lassen. Sehen Sie außerdem eine FOR-NEXT-Schleife vor, um die Lautstärke schrittweise, automatisch zu ändern.

....
....
....
....

— —

Unser Programm sieht so aus:

```
10 FOR L = 0 TO 15
20 SOUND 0,115,10,L
30 FOR W=1 TO 200: NEXT W
40 NEXT L
```

39. Für normale Anwendungen kann der Lautstärkewert ständig bei 8 bleiben. Als nächstes wollen wir die Ton-Variable ausprobieren. Bis jetzt haben wir den Wert 10 verwendet, jedoch kann die Variable N

Werte zwischen 0 und 15 annehmen. Ändern Sie unser Programm aus Abschnitt 38 so ab, daß jetzt T variiert wird. Setzen Sie die Lautstärke auf den Wert 8 fest.

....

....

....

....

— —

Das geänderte Programm:

```
10 FOR T=0 TO 15
20 SOUND 0,115,T,8
30 FOR W=1 TO 200: NEXT W
40 NEXT T
```

40. Wie würden Sie die durch das letzte Programm erzeugten Klänge beschreiben? ....

....

— —

Wir hörten einige schön klingende Töne, eine Reihe komischer Geräusche und schließlich angenehme Stille.

41. Lassen Sie uns jetzt den Ton wieder auf 10 festlegen, die Lautstärke belassen wir bei 8 und behalten "Voice 0" bei. Diesmal wollen wir die Tonhöhe ändern, die durch "Note" beeinflußt wird. Die vorgesehenen Werte reichen von 0 bis 255. Kürzen Sie die Zeitverzögerung im Programm aus Abschnitt 39 auf 50 und schreiben Sie das Programm so um, daß diesmal die Tonhöhen geändert werden.

....

....

....

....

— —

```
10 FOR N=0 TO 255
20 SOUND 0,N,10,8
30 FOR W=1 TO 50:NEXT W
40 NEXT N
```

42. Wenn Sie beabsichtigen, eine Melodie zu schreiben, werden Sie sicherlich nicht alle 256 Töne benötigen. Nachstehend finden Sie daher eine Tabelle mit den Werten von N für einen Tonumfang von 1 1/2 Oktaven. (T muß dabei auf 10 gesetzt werden).

2. Oktave

N	Ton
26	D#
27	D
29	C#
31	C
32	B
34	A#
36	A
38	G#
41	G
43	F#
46	F
48	E
51	D#
54	D
58	C#
61	C

1. Oktave

N	Ton
61	C
65	B
69	A#
73	A
77	G#
82	G
86	F#
92	F
97	E
103	D#
109	D
115	C#
122	C
129	B
137	A#
145	A

a) Welcher Wert aus der obigen Tabelle liefert, nach den Ergebnissen des Programms aus Abschnitt 41, den höchsten Ton? ....

b) Und den niedrigsten? ....

---

a) 26 b) 145

43. Um eine Tonleiter zu spielen, können wir Noten in einer Tabelle speichern und sie bei Bedarf lesen.

Tragen Sie im nachfolgenden Programm die Datenwerte ein. Verwenden Sie keine durch # erhöhten Töne und beginnen Sie mit dem niedrigsten Ton aus der Tabelle in Abschnitt 42.

```
10 FOR A=1 TO 16
20 READ N
30 SOUND 0,N,10,8
40 FOR W=1 TO 200: NEXT W
50 SOUND 0,0,10,8
60 FOR W=1 TO 10: NEXT W
70 NEXT A
80 DATA ......................................
90 DATA ......................................
100 DATA .....................................
```

```
80 DATA 145,129,122,109,97,92
90 DATA 82,73,65,61,54,48
100 DATA 46,41,36,32
```

44. Um Farben und Töne gleichzeitig zu erzeugen, fügen Sie bitte die folgenden Zeilen in das Programm aus Abschnitt 43 ein.

```
 5 GR. 5
15 COLOR 1
34 SETCOLOR 0,A–1,2
35 PLOT 5,11: DRAWTO 15,11
```

a) Wieviele Töne werden beim Lauf dieses Programms gespielt? ....
b) Wieviele Farben werden sichtbar? ....

---

a) 16 b) 16

45. Blau ist kalt, rot ist warm. Vervollständigen Sie bitte das folgende Programm, durch das 8 Töne gespielt werden. Kombinieren Sie warme Farben mit hohen Tönen und kalte Farben mit tiefen Tönen. Verwenden Sie die folgenden Farben:

Dunkelblau	Gelbgrün
Blau	Gelb
Blaugrün	Orange
Grün	Rot

Fügen Sie sowohl die Farben und die Töne in die DATA-Statements ein:

```
 10 GR.3
 20 FOR A=I TO 8
 30 COLOR 1
 40 READ N,C
 50 SETCOLOR 0, ..............................
 60 PLOT 10,9–A: DRAWTO 20,9–A
 70 SOUND 0, ..................., 10,8
 80 FOR W = 1 TO 200: NEXT W
 90 SETCOLOR 0,0,2
100 Sound 0,0,10,8
110 FOR W = 1 TO 10: NEXT W
120 NEXT A
130 DATA .................................
140 DATA .................................
150 DATA .................................
160 DATA .................................
```

Wir haben folgende Daten gewählt:

```
 50 SETCOLOR 0,C,2
 70 SOUND 0,N,10,8
130 DATA 122,9,109,10
140 DATA 97,13,92,15
150 DATA 82,1,73,2
160 DATA 65,3,61,5
```

46. Wenn Ihre Programmierfähigkeiten zunehmen, werden auch Ihre Programme immer länger werden. Längere Programme lassen sich aber wesentlich leichter schreiben und auch besser verstehen, wenn Sie in funktionelle Einheiten zerlegt werden. Oft werden Sie derartige Teilstücke an verschiedenen Stellen Ihres Programms verwenden. Statt Sie nun jedesmal zu wiederholen, kann eine SUBROUTINE verwendet werden. Das GOSUB-Statement weist den Computer an, das Haupt-Programm zu verlassen und zu dieser Subroutine zu springen. Sobald sie beendet ist, wird der Computer durch ein nachfolgendes RETURN-Statement aufgefordert, wieder in das Hauptprogramm zurückzukehren und in der nächsten auf das GOSUB-Statement folgenden Zeile fortzufahren.

Hier sehen Sie ein kurzes Programm, durch das die Verwendung der GOSUB- und RETURN-Statements demonstriert werden soll. Probieren Sie es einmal auf Ihrem Computer aus.

```
10 N = 122
20 GOSUB 100
30 N = 109
40 GOSUB 100
50 N = 97
60 GOSUB 100
70 END

10 SOUND 0,N,10,8
110 FOR W = 1 TO 200: NEXT W
120 SOUND 0,0,10,8
130 FOR W = 1 TO 10: NEXT W
140 RETURN
```

a) Welche Zeilen enthalten die Subroutine? ....

b) Wie oft wird die Subroutine verwendet? ....

c) Wieviele Noten werden gespielt? ....

— —

a) 100, 110, 120, 130, 140 b) 3 c) 3

47. Hier folgt nun ein Programm mit "verschachtelten" Subroutinen. In Zeile 110 wird die Subroutine 1000 aufgerufen. Anschließend wird in Zeile 1020, also noch innerhalb der ersten Subroutine, die Subroutine 2000 aufgerufen. Sie kehrt von Zeile 2010 zur Zeile 1030 der ersten Subroutine zurück. Daraufhin springt das Programm zurück zum Hauptprogramm. Das nachfolgende Programm spielt eine Tonleiter von einer Oktave und bildet gleichzeitig die Noten farbig auf dem Bildschirm ab.

```
1Ø GR. 3
2Ø COLOR 1
3Ø SETCOLOR 1,1Ø,2
4Ø FOR N = 5 TO 13 STEP 2
5Ø PLOT Ø,N: DRAWTO 38,N
6Ø PLOT Ø,15: DRAWTO 35,15
7Ø PLOT 3Ø,15: DRAWTO 35,15
8Ø GOSUB 2ØØØ
9Ø SETCOLOR 4,4,2
1ØØ COLOR 2
```

```
110 X=3: Y=15: N=122: GOSUB 1000
120 X=5: Y=14: N=109: GOSUB 1000
130 X=7: Y=13: N=97: GOSUB 1000
140 X=9: Y=12: N=92: GOSUB 1000
150 X=11: Y=11: N=82: GOSUB 1000
160 X=13: Y=10: N=73: GOSUB 1000
170 X=15: Y=9: N=65: GOSUB 1000
180 X=17: Y=8: N=61: GOSUB 1000
190 END

1000 PLOT X,Y
1010 SOUND 0,N,10,8
1020 GOSUB 2000
1030 RETURN

2000 FOR W= 1 TO 200: NEXT W
2010 RETURN
```

Ersetzen Sie Zeile 190 durch weitere Statements, um die Tonleiter wieder abwärts zu spielen und gleichzeitig die Noten darzustellen. (Bisher spielt das Programm die Tonleiter nur aufwärts.)

....
....
....
....
....
....
....
....

Hier sehen Sie eine Möglichkeit:

```
190 X=15: Y=9: N=65: GOSUB 1000
200 X=13: Y=10: N=73: GOSUB 1000
210 X=11: Y=11: N=82: GOSUB 1000
220 X=9: Y=12: N=92: GOSUB 1000
230 X=7: Y=13: N=97: GOSUB 1000
240 X=5: Y=14: N=109: GOSUB 1000
250 X=3: Y=15: N=122: GOSUB 1000
260 END
```

Das Tonerzeugungs-System Ihres ATARI-Computers ermöglicht die gleichzeitige Erzeugung von vier Stimmen (Voice), und zwar durch Variation von V im Sound-Statement.

SOUND V,N,T,L

Bis jetzt haben wir nur eine Stimme benutzt, aber das soll nun anders werden.

```
10 L0 = 8: L1 = 0
20 N0 = 41: N1 = 129
30 GOSUB 1000
40 L1 = 8: GOSUB 1000
50 L0 = 0: GOSUB 1000
60 L1 = 0: GOSUB 1000
70 STOP

1000 SOUND 0,N0,10,L0
1010 SOUND 1,N1,10,L1
1020 FOR W = 1 TO 100: NEXT W
1030 RETURN
```

a) Wieviele Töne hören Sie, wenn Zeile 30 ausgeführt wird? ....
b) Wieviele Töne sind bei der Ausführung von Zeile 40 zu hören? .....
c) Wieviele Töne hören Sie bei der Ausführung von Zeile 60? ....
d) Welche Stimme wird durch Zeile 50 abgeschaltet? ....

---

a) 1, da die Lautstärke von Voice 1 auf 0 gesetzt wurde.
b) 2 c) keinen d) Voice 0

49. Das folgende Programm erzeugt eine einfache Melodie mit zwei Stimmen. Um die einzelenen Töne jeweils in der richtigen Länge anzuhalten, werden drei Subroutinen verwendet. Jede Zeile des Haupt-Programms ruft eine Subroutine auf, um ein Notenpaar zu spielen.

```
10 N0 = 41: N1 = 129: GOSUB 2000
20 N0 = 32: N1 = 109: GOSUB 1000
30 N0 = 27: N1 = 92: GOSUB 3000
40 N0 = 31: N1 = 97: GOSUB 2000
50 N0 = 32: N1 = 109: GOSUB 1000
60 N0 = 36: N1 = 122: GOSUB 3000
70 N0 = 41: N1 = 129: GOSUB 1000
80 N0 = 36: N1 = 122: GOSUB 1000
90 N0 = 32: N1 = 109: GOSUB 1000
100 N0 = 36: N1 = 122: GOSUB 3000
110 N0 = 27: N1 = 92: GOSUB 2000
120 N0 = 0: N1 = 0: GOSUB 1000
130 N0 = 41: N1 = 129: GOSUB 2000
140 N0 = 32: N1 = 109: GOSUB 1000
150 N0 = 25: N1 = 92: GOSUB 3000
160 N0 = 36: N1 = 122: GOSUB 2000
170 N0 = 32: N1 = 109: GOSUB 1000
180 N0 = 31: N1 = 97: GOSUB 1000
190 N0 = 32: N1 = 109: GOSUB 1000
200 N0 = 36: N1 = 122: GOSUB 1000
210 N0 = 27: N1 = 92: GOSUB 2000
220 N0 = 41: N1 = 129: GOSUB 1000
```

```
230 NØ = 32: N1 = 1Ø9: GOSUB 2ØØØ
24Ø NØ = 36: N1 = 122: GOSUB 1ØØØ
25Ø NØ = 41: N1 = 129: GOSUB 3ØØØ
26Ø GOSUB 1ØØØ
27Ø GR. Ø
28Ø END
```

Füllen Sie in den folgenden Subroutinen die leeren Stellen aus:

1000 SOUND 0, ...
1010 SOUND 1, ...
1020 FOR W = 1 TO 50: NEXT W
1030 ....
2000 SOUND 0, ...
2010 SOUND 1, ...
2020 FOR W = 1 TO 100: NEXT W
2030 ....
3000 SOUND 0, ...
3010 SOUND 1, ...
3020 FOR W = 1 TO 150: NEXT W
3030 ....

— — — — — — — — — — — — — — — — — — — —

```
1ØØØ SOUND Ø,NØ,1Ø,8
1Ø1Ø SOUND 1,N1,1Ø,8
1Ø3Ø RETURN

2ØØØ SOUND Ø,NØ,1Ø,8
2Ø1Ø SOUND 1,N1,1Ø,8
2Ø3Ø RETURN

3ØØØ SOUND Ø,NØ,1Ø,8
3Ø1Ø SOUND 1,N1,1Ø,8
3Ø3Ø RETURN
```

50. Ändern Sie die Subroutinen im letzten Programm so ab, daß die Graphik-Betriebsart 3 verwendet wird. Stellen Sie in der Subroutine 1000 einen blauen, in der Subroutine 2000 einen goldenen und in der Subroutine 3000 einen roten Balken in der Nähe der Bildschirmmitte dar.

1011 ....
1012 ....
1013 ....
1014 ....

2011 ....
2012 ....
2013 ....
3011 ....
3012 ....
3013 ....
3014 ....

```
1Ø11 GR. 3
1Ø12 COLOR 1
1Ø13 SETCOLOR Ø,8,2
1Ø14 PLOT Ø,12: DRAWTO 5,12

2Ø11 GR. 3
2Ø12 COLOR 1
2Ø13 SETCOLOR Ø,1,2
2Ø14 PLOT Ø,12: DRAWTO 1Ø,12

3Ø11 GR. 3
3Ø12 COLOR 1
3Ø13 SETCOLOR Ø,4,2
3Ø14 PLOT Ø,12: DRAWTO 15,12
```

51. Das nächste Programm erstreckt sich über die Abschnitte 51 bis 55. Darin wollen wir zu unserer Melodie zwei–, drei– und vierstimmige Harmonien hinzufügen. Das Programm ist ziemlich lang. Daher soll es in mehrere Abschnitte unterteilt werden.

Eine der Stimmen spielt im Hintergrund eine sich wiederholende Folge von drei Tönen. Daher haben wir drei Subroutinen für die Sound-Statements verwendet, wobei jede Subroutine einen dieser Hintergrund-Töne enthält. Zusätzlich enthalten die Subroutinen aber auch die Sound-Statements für die beiden anderen Stimmen. Da diese Subroutinen die wichtigsten Bestandteile des Programmes sind, wollen wir uns mit Ihnen zuerst beschäftigen.

```
1000 SOUND 0,183,10,8          erster Hintergrund-Ton
1010 SOUND 1,N1,10,8           Voice 1
1020 SOUND 2,N2,10,8           Voice 2
1030 FOR W=1 TO 25:NEXT W      Notenlänge
1040 RETURN
2000 SOUND 0,137,10,8          zweiter Hintergrund-Ton
2010 SOUND 1,N1,10,8           Voice 1
2020 SOUND 2,N2,10,8           Voice 2
```

```
2030 FOR W=1 TO 25:NEXT W      Notenlänge
2040 RETURN
3000 SOUND 0,145,10,8          dritter Hintergrund-Ton
3010 SOUND 1,N1,10,8           Voice 1
3020 SOUND 2,N2,10,8           Voice 2
3030 FOR W=1 TO 25:NEXT W      Notenlänge
3040 RETURN
```

Jede dieser Subroutinen spielt gleichzeitig eine bestimmte Anzahl von Tönen für eine vorgegebene Zeitdauer.

a) Wieviele Töne gleichzeitig? ....

b) Welche Statements legen die Länge der einzelnen Noten fest? ......

---

a) 3   b) die FOR-NEXT-Schleifen in den Zeilen 1030, 2030 und 3030

52. Jetzt wollen wir uns den Anfang des Programms ansehen, in dem die Noten programmiert werden, die gespielt werden sollen.

```
10 N1=0:N2=0:GOSUB 1000      N1, N2 stumm
20 GOSUB 2000
30 GOSUB 3000
40 GOSUB 1000
50 GOSUB 2000
60 N2=61:GOSUB 3000          Voice 2 wird mit Ton 2 aktiviert
70 N2=46:GOSUB 1000          Ton 2 wird geändert
80 GOSUB 2000                gleicher Ton, aber Änderungen
90 GOSUB 3000                im Hintergrund
100 END
```

Geben Sie jetzt diesen Teil des Programms, zusammen mit den Subroutinen, ein und lassen Sie es ablaufen.

a) Wieviele Töne sind in Zeile 10 zu hören? ....

b) Ändert Zeile 20 den Klang? ....

c) Welche Zeile bewirkt, daß ein zweiter Ton zu hören ist? ....

---

a) Nur einer, der Hintergrundton (183)

b) Der Hintergrundton ändert sich in 137.

c) Zeile 60 (aber auch Zeile 70, 80, 90)

53. Ersetzen Sie Zeile 100 und fahren Sie wie folgt fort:

```
100 N1=48:N2=41:GOSUB 1000 Voice 1 und 2 ein
110 N1=46:N2=36:GOSUB 2000 ändert den Ton
120 N1=41:N2=34:GOSUB 3000 ändert den Ton
130 N1=36:N2=31:GOSUB 1000 ändert den Ton
140 GOSUB 2000             ändert nur den Hintergrund
150 GOSUB 3000             ändert nur den Hintergrund
160 N1=48:N2=41:GOSUB 1000 ändert alle drei Töne
170 GOSUB 2000             ändert nur den Hintergrund
180 GOSUB 3000             ändert erneut den Hintergrund
190 END
```

Lassen Sie nun auch dieses Programm ablaufen.

a) Wieviele Töne werden durch Zeile 100 veranlaßt? ....

b) Erzeugen sämtliche Zeilen von 100 – 180 jeweils drei Töne? ....

---

a) 3 b) ja

54. Der nächste Abschnitt fährt in der gleichen Weise fort, bis auf die letzten drei Zeilen (280 – 320).

```
190 N1=46:N2=36:GOSUB 1000
200 GOSUB 2000
210 N1=48:GOSUB 3000
220 N1=54:N2=34:GOSUB 1000
230 N1=61:N2=36:GOSUB 2000
240 N1=73:N2=46:GOSUB 1000
250 N1=73:N2=46:GOSUB 1000
260 GOSUB 2000
270 N1=69:N2=54:GOSUB 3000
280 SOUND 0,183,10,8
290 SOUND 1,82,10,8
300 SOUND 2,69,10,8
310 SOUND 3,61,10,8
320 FOR W=1 TO 50:NEXT W
```

a) Wieviele Stimmen sind während der Ausführung von Zeile 320 zu hören? ....

b) Wie verhält sich die zeitliche Länge der Töne zu der in den Subroutinen verwendeten Tondauer? ....

----

a) 4 b) Sie ist doppelt so groß (FOR W = 1 TO 50)

55. Hier ist der Rest des Programms.

```
330 N1=61:N2=0:N3=0:GOSUB 3000
340 N1=73:N2=46:GOSUB 1000
350 GOSUB 2000
360 N1=46:N2=0:GOSUB 3000
370 N1=61:N2=41:GOSUB 1000
380 N1=46:N2=36:GOSUB 2000
390 N1=41:N2=34:GOSUB 3000
400 N1=36:N2=31:GOSUB 1000
410 GOSUB 2000
420 N1=46:GOSUB 3000
430 N1=48:N2=41:GOSUB 1000
440 GOSUB 2000
450 N1=0:N2=36:GOSUB 3000
460 N1=54:N2=34:GOSUB 1000
470 GOSUB 2000
480 N1=0:GOSUB 3000
490 N1=61:N2=34:GOSUB 1000
500 N2=36:GOSUB 2000
510 N1=69:N2=41:GOSUB 3000
520 N1=73:N2=61:GOSUB 1000
530 N1=46:GOSUB 2000
540 N1=82:N2=61:GOSUB 3000
550 SOUND 0,183,10,8
560 SOUND 1,92,10,8
570 SOUND 2,46,10,8
580 FOR W=1 TO 75:NEXT W
590 END
```

a) Wievie Noten enthielt der letzte gespielte Ton? ....
b) Wie lang war die Dauer des letzten Tons im Vergleich mit der in den Subroutinen verwendeten Tondauer? ....

----

a) 3 b) dreifache Länge

EIGENTEST

Unseren Grückwunsch!!! In diesem Kapitel haben Sie wieder eine ganze Menge gelernt. Wir haben Ihnen einige Grundkenntnisse vermittelt, auf denen Sie aufbauen können um die Graphik- und Klangerzeugungs-Möglichkeiten Ihres ATARI-Computers noch besser kennenzulernen. Verwenden Sie diesen Eigentest dazu, um Ihr Wissen zu demonstrieren.

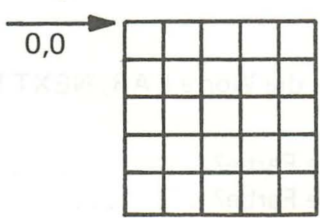
1. Bei Ihrem ATARI-Computer sind 8 verschiedene Bildschirm-Betriebsarten zum Mischen von Text und Graphiken vorgesehen. Sie werden von 0 bis 7 numeriert. In den früheren Kapiteln wurden alle Programme in der Text-Betriebsart geschrieben.

   a) Welches Graphik-Statement wird für die Textdarstellung verwendet? ....

   b) Geben Sie eine gemischte Graphik-Text-Betriebsart mit ungefähr 160x80 Graphik-Positionen an. ....

2. Schraffieren Sie in dem abgebildeten Raster die Punkte, die durch folgendes Statement bezeichnet werden.

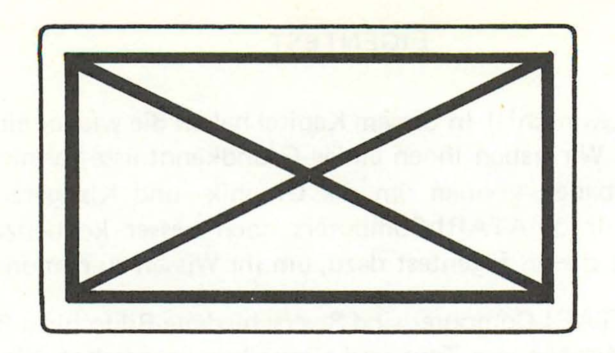
   20 PLOT 4,1



3. Vervollständigen Sie das nachfolgende Programm so, daß es eine schräge Linie von der linken oberen Ecke in die rechte untere Ecke des Graphikfeldes zeichnet.

   10 GRAPHICS 4
   20 COLOR 1
   30 PLOT ....
   40 DRAWTO ....

4. Schreiben Sie für den Graphik-Mode 6 ein Programm, welches das größtmögliche Rechteck in die Graphikfläche zeichnet. Ziehen Sie auch, wie in der Skizze angedeutet, die Diagonalen.



5. Lassen Sie das folgende Programm auf Ihrem Computer ablaufen, um die Fragen 5 a), b) und c) zu beantworten.

```
10 GRAPHICS 3
20 COLOR 1
30 FOR M=0 TO 4 STEP 2
40 FOR N=0 TO 15
50 SETCOLOR M,N,2
60 PLOT 2,10:DRAWTO 12,10
70 FOR W=1 TO 200:NEXT W
80 NEXT N
90 NEXT M
```

Schreiben Sie in die leeren Stellen die Worte BAR, NEXT WINDOW oder BACKGROUND.

a) Welches Wort ändert bei M=0 die Farbe? ....
b) Welches Wort ändert bei M=2 die Farbe? ....
c) Welches Wort ändert bei M=4 die Farbe? ....

6. Was geschieht auf dem Bildschirm, wenn das folgende Programm abläuft? ....

```
10 GRAPHICS 3
20 COLOR 1
30 FOR N=0 TO 15
40 SETCOLOR 0,N+3,2
50 PLOT 2,5:DRAWTO 2,10
60 FOR W=1 TO 200:NEXT W
70 SETCOLOR 4,N,2
80 FOR W=1 TO 200:NEXT W
90 NEXT N
```

7. Schreiben Sie ein Programm, durch das die Zeichenfläche im Graphik-Mode 3 mit abwechselnden Punkten gefüllt wird (d. h. 0,0; 0,2; 0,4 . . . 2,0; 2,2 . . .). Benutzen Sie dazu verschachtelte FOR-NEXT-Schleifen und programmieren Sie Ihre Lieblingsfarben.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

8. Jetzt wollen wir das SOUND-Statement benutzen.

a) Wieviele verschiedene Stimmen können gleichzeitig gespielt werden? . . . .

b) Die Lautstärkewerte können auf jede ganze Zahl zwischen . . . . und . . . . gesetzt werden.

c) Die Tonhöhen-Variable N kann von . . . . bis . . . . reichen.

9. Das SOUND-Statement im folgenden Programm veranlaßt, daß ein Ton hörbar wird. Wie lange erklingt er? . . . .

```
10 INPUT N
20 SOUND 0,97,10,8
30 GOTO 10
```

10. Schreiben Sie, unter Verwendung der Tabelle in Abschnitt 42 ein Programm, durch das eine Oktave der C-Dur-Tonleiter (C bis C) gespielt wird.

. . . .
. . . .
. . . .
. . . .
. . . .
. . . .
. . . .

11. Gegeben ist das Programm:

```
10 N0=36: N1=122: GOSUB 1000
```

20 N0=32: N1=109: GOSUB 2000
30 END

Schreiben Sie die Subroutinen, welche die Töne spielen. Die Subroutine 1000 sollte Töne mit einer Länge von 50 Einheiten und Subroutine 2000 solche mit einer Länge von 100 Einheiten erzeugen.

....

....

....

....

....

....

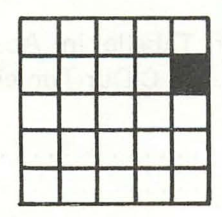
....

....

Antworten zum Eigentest

1. a) 10 GRAPHICS 0 (Abschnitt 4)
   b) 10 GRAPHICS 6 oder 10 GRAPHICS 7 (Abschnitt 5)

2. 20 PLOT 4,1 bezeichnet das eingezeichnete Rechteck in der fünften Spalte und der zweiten Reihe.



3. 30 PLOT 0,0
   40 DRAWTO 78,39

4.
```
10 GRAPHICS 6
20 COLOR 1
30 SETCOLOR 0,12,2
40 PLOT 0,0
50 DRAWTO 157,0
60 DRAWTO 157,79
70 DRAWTO 0,79
```

```
80 DRAWTO 0,0
90 DRAWTO 157,79
100 PLOT 0,79
110 DRAWTO 157,0
```

(Abschnitte 5,7,18,20)

5. a) BAR (Abschnitte 23,26)
   b) NEXT WINDOW (Abschnitte 24,26)
   c) BACKGROUND (Abschnitte 25,26)

6. Sowohl BAR als auch BACKGROUND ändern die Farbe. (Abschnitte 23 – 26)

7. Unser Programm:

```
10 GRAPHICS 3
20 COLOR 1
30 SETCOLOR 0,2,2
40 FOR N = 0 TO 39 STEP 2
50 FOR M = 0 TO 19 STEP 2
60 PLOT N,M
70 NEXT M
80 NEXT N
```

Abschnitte 5,9,23)

8. a) 4 (Abschnitt 47)
   b) 0 bis 15 (Abschnitt 37)
   c) 0 bis 255 (Abschnitt 41)

9. Solange, bis ein neuer Ton durch INPUT eingegeben wird. (Abschnitt 38)

10. Ihre Programm sollte etwa so aussehen:

```
10 FOR A = 1 TO 8
20 READ N
30 SOUND 0,N,10,8
40 FOR W = 1 TO 200: NEXT W
50 NEXT A
60 DATA 122,109,97,92
70 DATA 82,73,65,61
```

Die Zeilen 60 und 70 können auch lauten:

```
60 DATA 61,54,48,46
70 DATA 41,36,32,31
```

(Abschnitt 42)

11.
```
1000 SOUND 0,N0,10,8
1010 SOUND 1,N1,10,8
1020 FOR W = 1 TO 50: NEXT W
1030 RETURN

2000 SOUND 0,N0,10,8
2010 SOUND 1,N1,10,8
2020 FOR W = 1 TO 100: NEXT W
2030 RETURN
```

NOTIZEN

ABSCHLUSS–TEST

Nachdem Sie nun das ganze Buch durchgearbeitet haben, können Sie alles, was Sie bisher gelernt haben, im folgenden Abschlußtest noch einmal überprüfen. Die jeweiligen Lösungen finen Sie im Anschluß an den Test.

1. "STARS" ist der Name eines Zahlenratespiels, das zuerst in der Zeitschrift "People's Computer Company" veröffentlicht wurde. Es ist für Spieler aller Altersgruppen geeignet, aber auch seine Programmierung ist eine reizvolle Aufgabe.

   Hier finden Sie die Regeln, die der Computer auf dem Bildschirm ausgibt. Sie brauchen sie natürlich nicht mehr in Ihrem Programm vorzusehen.

   > "Ich denke mir eine ganze Zahl zwischen 1 und 100. Versuchen Sie, meine Zahl zu erraten. Wenn Sie einen Rateversuch gemacht haben, werde ich einen oder mehr Sterne (*) abbilden. Je näher Sie an meiner Zahl sind, um so mehr Sterne werde ich abbilden. Ein Stern (*) bedeutet, daß Sie von meiner Zahl sehr weit entfernt sind. Sieben Sterne sagen Ihnen, daß Sie sehr, sehr nahe an meiner Zahl sind.!!!"

   Die Logik dieses Spiels sieht folgendermaßen aus. Wenn Ihre Zahl um mehr als 64 von der richtigen entfernt liegt, wird ein Stern abgebildet; 32 . . . 63 entfernt: 2 Sterne; 16 . . . 31 entfernt: 3 Sterne; 8 . . . 15 enfernt: 4 Sterne; 4 . . . 7 entfernt: 5 Sterne; 2 . . . 3 entfernt: 6 Sterne; bei einer Abweichung von nur 1: 7 Sterne. Zur Programmierung benötigen Sie die Absolut-Wert-Funktion. Die für Sie zwar noch neu, aber trotzdem leicht anzuwenden ist.

   Beispiele:
   ABS(10)=10
   ABS(–10)=10
   IF ABS(X–Y)= 10 THEN 100

   Ein Lauf sollte so aussehen:

   ```
   RUN
   GEBEN SIE IHRE ZAHL EIN? 10
   **
   ```

```
GEBEN SIE IHRE ZAHL EIN? 50
******

GEBEN SIE IHRE ZAHL EIN? 48
SIE HABEN GEWONNEN!!!
```

2. Vielleicht haben Sie schon auf einem anderen Computer Matrix-Befehle verwendet. (Matrix ist ein anderes Wort für Array). Die bei Heim-Computern verwendeten BASIC-Versionen enthalten meist keine Matrix-Kommandos, jedoch sind sie leicht zu simulieren.

   Versuchen Sie das einmal an folgendem Beispiel. Sie haben zwei 5x3-Arrays A und B, die mit kleinen Zahlen gefüllt sind. Diese Zahlen werden aus DATA-Statements in die Elemente des Arrays eingelesen. Schreiben Sie nun ein Programm, um jedes Element des Arrays zu dem entsprechenden Array aus Element B zu addieren und die Summe an die gleiche Position in einem dritten Array C einzuschreiben.

A

7	8	9
3	4	1
6	8	2
1	4	1
8	10	2

B

3	6	8
2	4	7
1	3	1
8	4	9
6	6	6

C

10	14	17
5	8	8
7	11	3
9	8	10
14	16	8

Beispiel:
A(1,1) + B(1,1) = C(1,1)
A(1,2) + B(1,2) = C(1,2) usw.

```
RUN
INHALT DES A–ARRAYS
 7        8        9
 3        4        1
 6        8        2
 1        4        1
 8       10        2
INHALT DES B–ARRAYS
 3        6        8
 2        4        7
```

1	3	1
8	4	9
6	6	6

INHALT DES C–ARRAYS

10	14	17
5	8	8
7	11	3
9	8	10
14	16	8

3. Schreiben Sie ein Programm, durch das amerikanische Maße in metrische und umgekehrt umgewandelt werden. Schreiben Sie Ihr Programm so, daß der Benutzer die gewünschte Umwandlung auswählen kann, wie es der folgende Lauf zeigt.

```
RUN
WUENSCHEN SIE EINE UMWANDLUNG VON
(1) METRISCHEN IN AMERIKANISCHE ODER
(2) AMERIKANISCHEN IN METRISCHE EINHEITEN
GEBEN SIE 1 ODER 2 EIN? 1
WUENSCHEN SIE EINE LISTE DER UMWANDLUNGEN (J ODER N)? J
WAEHLEN SIE DIE UMWANDLUNG AUS DIESER LISTE:
(1) ZENTIMETER IN ZOLL
(2) METER IN FUSS
(3) KILOMETER IN MEILEN
(4) KILOGRAMM IN POUND
(5) GRAMM IN UNZEN
(6) LITER IN QUARTS
(7) GRAD CELSIUS IN GRAD FAHRENHEIT
GEBEN SIE DIE NUMMER DER GEWUENSCHTEN UMWANDLUNG EIN? 5
WIEVIEL GRAMM? 10
  10 GRAMM = .35 UNZEN
WUENSCHEN SIE EINE UMWANDLUNG VON
(1) METRISCHEN IN AMERIKANISCHE ODER
(2) AMERIKANISCHEN IN METRISCHE EINHEITEN
GEBEN SIE 1 ODER 2 EIN? 2
WUENSCHEN SIE EINE LISTE DER UMWANDLUNGEN (J ODER N)? J
WAEHLEN SIE DIE UMWANDLUNG AUS DIESER LISTE:
(1) ZOLL IN ZENTIMETER
(2) FUSS IN METER
(3) MEILEN IN KILOMETER
```

```
(4) POUND IN KILOGRAMM
(5) UNZEN IN GRAMM
(6) QUARTS IN LITER
(7) GRAD FAHRENHEIT IN GRAD CELSIUS
GEBEN SIE DIE NUMMER DER GEWUENSCHTEN UMWANDLUNG EIN? 3
WIEVIELE MEILEN? 2
  2 MEILEN = 3.218 KILOMETER
WUENSCHEN SIE EINE UMWANDLUNG VON
(1) METRISCHEN IN AMERIKANISCHE ODER
(2) AMERIKANISCHEN IN METRISCHE EINHEITEN
GEBEN SIE 1 ODER 2 EIN? 1
WUENSCHEN SIE EINE LISTE DER UMWANDLUNGEN (J ODER N)? N
GEBEN SIE DIE NUMMER DER GEWUENSCHTEN UMWANDLUNG EIN? 1
WIEVIEL ZENTIMETER? 50
  50 ZENTIMETER = 19.5 ZOLL
```

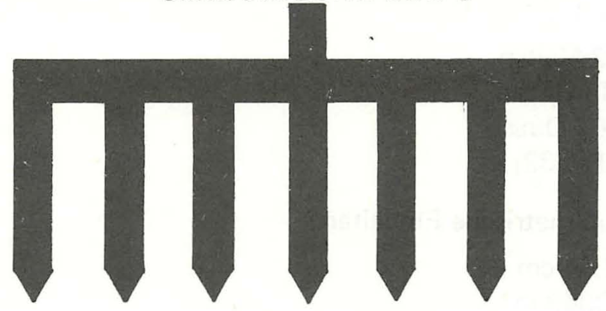
usw.

Wir schlagen Ihnen vor, zur Organisation dieses Programms umfangreichen Gebrauch von Subroutinen zu machen. Das Hauptprogramm sehen Sie nachfolgend. In diesem Segment wählt der Benutzer aus, welche Art von Umwandlungen er wünscht. Das Programm verzweigt dann zu Subroutinen, welche die speziellen Umwandlungen in Listen enthalten. Von diesen beiden Subroutinen aus verzweigt das Programm nochmals zu einer von vierzehn Sub-Subroutinen (sieben für jede Umwandlungs-"Richtung").

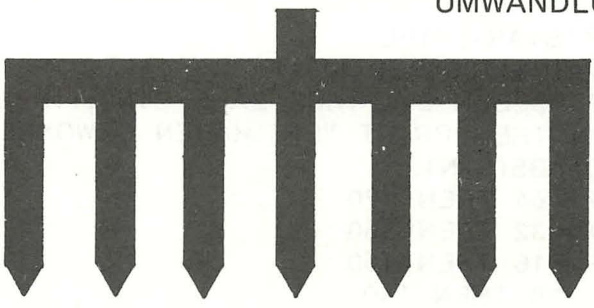
```
WUENSCHEN SIE EINE UMWANDLUNG VON
(1) METRISCHEN IN AMERIKANISCHE ODER
(2) AMERIKANISCHEN IN METRISCHE EINHEITEN
GEBEN SIE 1 ODER 2 EIN

WUENSCHEN SIE EINE LISTE DER UMWANDLUNGEN
                                   (J ODER N)? J
WAEHLEN SIE DIE UMWANDLUNG AUS DIESER LISTE:
(1) ZENTIMETER IN ZOLL
(2) METER IN FUSS
(3) KILOMETER IN MEILEN
(4) KILOGRAMM IN POUND
(5) GRAMM IN UNZEN
```

(6) LITER IN QUARTS
(7) GRAD CLSIUS IN GRAD FAHRENHEIT
GEBEN SIE DIE NUMMER DER GEWUENSCHTEN
UMWANDLUNG EIN? 5



WUENSCHEN SIE EINE LISTE DER UMWANDLUNGEN
(J ODER N) J
WAEHLEN SIE DIE UMWANDLUNG AUS DIESER LISTE:
(1) ZOLL IN ZENTIMETER
(2) FUSS IN METER
(3) MEILEN IN KILOMETER
(4) POUND IN KILOGRAMM
(5) UNZEN IN GRAMM
(6) QUARTS IN LITER
(7) GRAD FAHRENHEIT IN GRAD CELSIUS
GEBEN SIE DIE NUMMER DER GEWUENSCHTEN
UMWANDLUNG EIN



Kehren Sie aus jeder dieser Subroutinen direkt zu der Subroutine zurück, von der aus der GOSUB-Befehl erfolgte. Von dort springen Sie unmittelbar, für eine neue Umwandlung, wieder ins Hauptprogramm zurück.

Nachfolgend finden Sie die zur Umrechnung benötigten Angaben:

Metrische in US-Einheiten

1 cm = .39 Zoll
1 m = 3,28 Fuß
1 km = .62 Meilen
1 g = 0.035 Unzen
1 l = 1,0567 Quarts
°C = 5/9(F° -32)

US-Maße in metrische Einheiten

1 Zoll = 2,54 cm
1 Fuß = .3048 m
1 Meile = 1,609 km
1 Pound = 0,45 kg
1 Quart = 0,946 l
F° = 9/5 °C +32
1 kg = 2,2 Pound

Antworten zum Abschlußtest

1.

```
10 REM***STAR-SPIEL
20 LET N=INT(100*RND(1))+1
30 PRINT "GEBEN SIE IHRE ZAHL EIN";:INPUT G
35 IF G=N THEN PRINT "SIE HABEN GEWONNEN!!!":GOTO 20
40 LET D=ABS(G-N)
50 IF D>=64 THEN 170
60 IF D>=32 THEN 160
70 IF D>=16 THEN 150
80 IF D>=8 THEN 140
90 IF D>=4 THEN 130
100 IF D>=2 THEN 120
110 PRINT "*";
120 PRINT "*";
130 PRINT "*";
140 PRINT "*";
```

```
150 PRINT "*";
160 PRINT "*";
170 PRINT "*" : GOTO 30
```

2.

```
10 REM ***SIMULATION DER MATRIX-ADDITION
15 REM ***INITIALISIERUNG
20 DIM A(5,3),B(5,3),C(5,3)
25 REM ***LADE ARRAY A UND B AUS DEN
26 REM ***DATA-STATEMENTS, DRUECKE DIE ARRAYS
27 PRINT "INHALT VON ARRAY A":PRINT
30 FOR X=1 TO 5
35 FOR Y=1 TO 3
40 READ A:A(X,Y)=A:PRINT A(X,Y),
45 NEXT Y
50 PRINT
55 NEXT X:PRINT :PRINT
57 PRINT "INHALT VON ARRAY B":PRINT
60 FOR X=1 TO 5
65 FOR Y=1 TO 3
70 READ B:B(X,Y)=B:PRINT B(X,Y),
75 NEXT Y
80 PRINT
85 NEXT X:PRINT :PRINT
90 REM ***ADDIERE A + B IN C
92 PRINT "INHALT VON ARRAY C":PRINT
95 FOR X=1 TO 5
100 FOR Y=1 TO 3
105 LET C(X,Y)=A(X,Y)+B(X,Y):PRINT C(X,Y),
110 NEXT Y
115 PRINT
120 NEXT X
130 REM ***DATEN FUER ARRAY A
135 DATA 7,8,9,3,4,1,6,8,2,1,4,1,8,10,2
140 REM ***DATEN FUER ARRAY B
145 DATA 3,6,8,2,4,7,1,3,1,8,4,9,6,6,6
```

```
100 REM ***UMWANDLUNG AUSGEWAEHLTER MASSEINHEITEN
105 DIM A$(1)
110 PRINT "WUENSCHEN SIE EINE UMWANDLUNG VON"
120 PRINT "(1) METRISCHEN IN AMERIKANISCHE ODER"
130 PRINT "(2) AMERIKANISCHEN IN METRISCHE EINHEITEN"
140 PRINT "GEBEN SIE 1 ODER 2 EIN";:INPUT N
150 ON N GOSUB 1000,1100
160 GOTO 110


1000 REM ***ANWENDER WAEHLT DIE GEWUENSCHTE UMWANDLUNG METRISCH/US
1010 PRINT "WUENSCHEN SIE EINE LISTE DER UMWANDLUNGEN (J ODER N)";:INPUT A$
1020 IF A$="N" THEN 1040
1030 PRINT "WAEHLEN SIE DIE UMWANDLUNG AUS DIESER LISTE:"
1031 PRINT "(1) ZENTIMETER IN ZOLL"
1032 PRINT "(2) METER IN FUSS"
1033 PRINT "(3) KILOMETER IN MEILEN"
1034 PRINT "(4) KILOGRAMM IN POUND"
1035 PRINT "(5) GRAMM IN UNZEN"
1036 PRINT "(6) LITER IN QUARTS"
1037 PRINT "(7) GRAD CELSIUS IN GRAD FAHRENHEIT"
1038 PRINT "(8) BEENDEN"
1040 PRINT "GEBEN SIE DIE NUMMER DER GEWUENSCHTEN UMWANDLUNG EIN";:INPUT M
1050 ON M GOSUB 2100,2200,2300,2400,2500,2600,2700,2800
1060 RETURN

2100 REM ***ZENTIMETER IN ZOLL
2110 PRINT "WIEVIEL ZENTIMETER";:INPUT C
2120 PRINT C;" ZENTIMETER = ";C*0.39;" ZOLL"
2130 PRINT :RETURN

2200 REM ***METER IN FUSS
2210 PRINT "WIEVIEL METER";:INPUT M
2220 PRINT M;" METER = ";M*3.28;" FUSS"
2230 PRINT :RETURN

2400 REM ***KILOGRAMM IN POUND
2410 PRINT "WIEVIEL KILOGRAMM";:INPUT K
2420 PRINT K;" KILOGRAMM = ";K*2.2;" POUND"
2430 PRINT :RETURN

2500 REM ***GRAMM IN UNZEN
2510 PRINT "WIEVIEL GRAM";:INPUT G
2520 PRINT G;" GRAMM = ";G*0.035;" UNZEN"
2530 PRINT :RETURN

2600 REM ***LITER IN QUARTS
2610 PRINT "WIEVIEL LITER";:INPUT L
2620 PRINT L;" LITER = ";L*1.0567;" QUARTS"
2630 PRINT :RETURN

2700 REM ***GRAD CELSIUS IN GRAD FAHRENHEIT
2710 PRINT "WIEVIEL GRAD CELSIUS";:INPUT D
2720 PRINT D;" GRAD CELSIUS = ";9*D/5+32;" GRAD FAHRENHEIT"
2730 PRINT :RETURN

2800 END
```

```
3100 REM ***ZOLL IN ZENTIMETER
3110 PRINT "WIEVIEL ZOLL";:INPUT I
3120 PRINT I;" ZOLL = ";I*2.54;" ZENTIMETER"
3130 PRINT :RETURN

3200 REM ***FUSS IN METER
3210 PRINT "WIEVIEL FUSS";:INPUT F
3220 PRINT F;" FUSS = ";F*0.3048;" METER"
3230 PRINT :RETURN

3300 REM ***MEILEN IN KILOMETER
3310 PRINT "WIEVIEL MEILEN";:INPUT M
3320 PRINT M;" MEILEN = ";M*1.609;" KILOMETER"
3330 PRINT :RETURN

3400 REM ***POUND IN KILOGRAMM
3410 PRINT "WIEVIEL POUND";:INPUT P
3420 PRINT P;" POUND = ";P*0.45;" KILOGRAMM"
3430 PRINT :RETURN

3500 REM ***UNZEN IN GRAMM
3510 PRINT "WIEVIEL UNZEN";:INPUT O
3520 PRINT O;" UNZEN = ";O*28.35;" GRAMM"
3530 PRINT :RETURN

3600 REM ***QUARTS IN LITER
3610 PRINT "WIEVIEL QUARTS";:INPUT Q
3620 PRINT Q;" QUARTS = ";Q*0.946;" LITER"
3630 PRINT :RETURN

3700 REM ***GRAD F IN GRAD C
3710 PRINT "WIEVIEL GRAD FAHRENHEIT";:INPUT D
3720 PRINT D;" GRAD FAHRENHEIT = ";5/9*(D-32);" GRAD CELSIUS"
3730 PRINT :RETURN
```

NOTIZEN

ANHANG

BASIC–FUNKTION

Dieser Anhang beschreibt einige der gebräuchlichsten und nützlichsten Funktionen, die sich bei den meisten, bei Personal-Computern üblichen, BASIC-Versionen finden. Diese Aufstellung ist keineswegs erschöpfend und darüber hinaus können auch Unterschiede in der Wirkung der Funktionen bei Ihrem Computer sein. Sehen Sie sich im Handbuch die komplette Liste der in Ihrer BASIC-Version zur Verfügung stehenden Funktionen an.

Arithmetische Funktionen

Im folgenden Text steht "EXP" für jeden beliebigen BASIC-Ausdruck, eine BASIC-Variable oder einen Zahlenwert.

ABS(exp): Liefert den Absolutwert des Ausdrucks, d. h., ABS(A)=A wenn A ⟩ = 0, ABS(1)=–A wenn A ⟨ 0.

Beispiel: IF ABS(X–G) ⟩ = 64 THEN PRINT "*"

EXP(exp): Berechnet die Exponential-Funktion zur Basis e, wobei e = 2,71828 . . . ist. Üblicherweise existiert eine obere Grenze für den Wert der Variablen. In BASIC muß der Wert in Klammern kleiner als 87.3365 sein.

Beispiel: B=EXP(X)
Y=C*EXP(–A*T)

FRE(0): Diese Funktion wird zur Bestimmung des freien Speicherplatzes, der nicht von BASIC oder vom Programm belegt ist, verwendet. Diese Funktion wird in einem PRINT-Statement eingegeben, mit einer in Klammern gesetzten 0. Der Computer gibt dann die Anzahl der unbelegten Bytes an.

Beispiel: PRINT FRE(0)

Siehe auch: FRE(String Variable) unter String-Funktionen

INT(exp): Berechnet die größte ganze Zahl, die kleiner oder gleich exp. ist. Beachten Sie, daß INT(3.14)=3, INT(–3.14) aber –4 ist. Sehen Sie sich in Ihrem Manual die sehr ähnliche Funktion FIX an.

LOG(exp): Liefert den natürlichen Logarithmus von exp. (Basis e).

Beispiel: D=LOG(1 + X ↑ 2)

RND(Parameter): Erzeugt, gesteuert durch den Parameterwert, eine Zufallszahl zwischen 0 und 1. In BASIC liefert jeder positive Wert in Klammern jedesmal eine neue Zufallszahl. Bei einer Null in der Klammer erhält man die Zufallszahl, die als letzte erzeugt wurde. Auch bei Verwendung negativer Zahlen man für jede Zahl eine andere Zufallszahl, jedoch erhält man für den gleichen negativen Klammerwert immer dieselbe Zufallszahl. Da die RND-Funktion bei Ihrem Computer anders wirken kann, lesen Sie bitte in Ihrem Handbuch nach.

SGN(exp): Liefert eine 1, wenn der Ausdruck einen positiven Wert hat (〉0); sie liefert eine 0, wenn der Ausdruck den Wert 0 hat und wird –1 für negative Werte des Ausdrucks.

Beispiel: ON SGN(X)+2 GOTO 100,200,300

Ist X negativ, verzweigt das obige Statement zur Zeile 100; ist X=0 erfolgt die Verzweigung zur Zeile 200; ist X positiv, verzweigt das Programm zur Zeile 300.

SQR(exp): Liefert die positive Quadratwurzel des Ausdrucks. Der Ausdruck in Klammern muß 0 sein oder einen positiven Wert haben.

Trigonometrische Funktionen

SIN(exp), COS(exp), TAN(exp), ATN(exp): die Funktion SIN, COS und TAN berechnen den Sinus, Cosinus oder den Tangens eines Ausdruckes, wobei der Winkel im Bogenmaß (rad) angegeben sein muß. ATN berechnet den Hauptwert des Arcustangens und zwar ebenfalls im Bogenmaß. Der Wert für ATN(exp) liegt im Bereich: $-\pi/2 < \text{ATN(exp)} < \pi/2$.

String–Funktion

ASC(String): Liefert den numerischen Wert des ASCII-Codes für das erste Zeichen eines Strings.

Beispiel: PRINT ASC(B$)

CHR$(exp): Wandelt eine Zahl in das entsprechende ASCII-Zeichen um.

Beispiel: Die ASCII-Code-Nummer 7 läßt bei einem Fernschreiber die Glocke, bzw. bei manchen CRT-Terminals einen Pieps ertönen. Um die Glocke zu läuten, müssen Sie CHR$(7) in einem PRINT-Statement vorsehen: z. B.: PRINT CHR$(7)

FRE(String Variable): Diese Funktion liefert die Anzahl der freien Bytes in dem für Strings reservierten Speicherbereich des Computers.

LEN(String): Gibt die Anzahl der in einem String enthaltenen Zeichen an; Leertasten werden als Zeichen gezählt.

Beispiel: FOR K–1 TO LEN(X$)

STR$(exp): Wandelt einen numerischen Ausdruck in einen String um. Das Minuszeichen eines negativen Wertes und die Leerstelle für ein "+" bei einem positiven Wert sind im String enthalten.

Beispiel: B$=STR$(X*Y)

VAL(String): Wandelt die String-Darstellung einer Zahl in einen numerischen Wert um. Der String muß ein numerisches Zeichen enthalten.

Beispiel: X$="33.3"
X=VAL(X$)

ASCII-Zeichen-Code

Dezimal	Zeichen	Dezimal	Zeichen
000	NUL	037	%
001	SO11	038	&
002	STX	039	’
003	ETX	040	(
004	EOT	041	)
005	ENQ	042	*
006	ACK	043	+
007	BEL	044	‘
008	BS	045	–
009	HT	046	.
010	LF	047	/
011	VT	048	0
012	FF	049	1
013	CR	050	2
014	SO	051	3
015	SI	052	4
016	DLE	053	5
017	DC1	054	6
018	DC2	055	7
019	DC3	056	8
020	DC4	057	9
021	NAK	058	:
022	SYN	059	;
023	ETB	060	<
024	CAN	061	=
025	EM	062	>
026	SUB	063	?
027	ESCAPE	064	@
028	FS	065	A
029	GS	066	B
030	RS	067	C
031	US	068	D
032	SPACE	069	E
033	!	070	F
034	“	071	G
035	#	072	H
036	$	073	I

Dezimal	Zeichen
074	J
075	K
076	L
077	M
078	N
079	O
080	P
081	Q
082	R
083	S
084	T
085	U
086	V
087	W
088	X
089	Y
090	Z
091	[
092	\
093	]
094	↑
095	←
096	`
097	a
098	b
099	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r

Dezimal	Zeichen
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	¦
125	}
126	~
127	DEL

Erläuterungen:
LF = Zeilen-Vorschub (Line Feed)
FF = Formularvorschub (Form Feed)
CR = Wagen-Rücklauf (Carriage Return)
DEL = Löschen (Delete)

FEHLER–MITTEILUNGEN

Fehler verursachen eine Mitteilung der Form:

"ERROR xx at LINE nnn."

Die Fehler Nummer ("xx" in der obigen Zeile kann im Bereich von 1 bis 199 liegen.

Fehler-Nummer 1:	POWER NOT ON
Fehler-Nummer 2:	MEMORY FULL
Fehler-Nummer 3:	VALUE ERROR
Fehler-Nummer 4:	VARIABLE TABLE FULL (zuviele Variablen)
Fehler-Nummer 5:	STRING LENGTH ERROR
Fehler-Nummer 6:	READ OUT OF DATA
Fehler-Nummer 7:	VALUE NOT + INTEGER
Fehler-Nummer 8:	INPUT STMT ERROR
Fehler-Nummer 9:	ARRAY/STRING DIM ERROR
Fehler-Nummer 10:	ARG STACK OVERFLOW
Fehler-Nummer 11:	FLOATING POINT OVERFLOW
Fehler-Nummer 12:	LINE NOT FOUND (GOSUB/GOTO)
Fehler-Nummer 13:	NO MATCHING FOR
Fehler-Nummer 14:	LINE TOO LONG
Fehler-Nummer 15:	GOSUB/FOR LINE DELETED
Fehler-Nummer 16:	BAD RETURNS
Fehler-Nummer 17:	EXECUTION OF GARBAGE
Fehler-Nummer 18:	STRING DOES NOT START WITH VALID NUMB
Fehler-Nummer 19:	LOAD PGM TOO BIG
Fehler-Nummer 20:	DEVICE NUMBER GREATER THAN 7
Fehler-Nummer 21:	NOT A LOAD FILE

Liebe Leser,

Wer bis hierhin das Buch durchgearbeitet hat, musste sicher viel Energie aufbringen. Dafür aber kann er nun mit Stolz von sich behaupten, eine neue Sprache gelernt zu haben. Die grösste Hürde ist genommen, und nun steht nichts mehr im Wege, den Dialog mit unserem Atari nach eigenen Wünschen zu gestalten.

"Halt", werden Sie sagen, "ein kleines aber steht hier noch im Raum". Richtig! Dieses Buch wurde in Amerika geschrieben und behandelt die dort erhältliche Version Ihres Ataricomputer.

Sie jedoch haben die "bessere Maschine". Ihr Atari 400/800 unterscheidet sich von der amerikanischen Ausführung nicht nur durch die Fernsehnorm (hier PAL, dort NTSC).

In seinem Inneren verbirgt sich ein elektronischer Baustein, GTIA genannt. Dieser Baustein, komplex wie ein Mikrocomputerbaustein, hat eine besondere Aufgabe. Immer dann, wenn wir in den verschiedenen Graphikstufen mit unterschiedlichen Farben arbeiten, benötigen wir seine Hilfe. Er prüft jeden einzelnen "PRINT" oder "PLOT" Befehl und markiert ihn mit unserem Farbwunsch. Er ist der "Generalmanager" der Farbdarstellung.

Hier kommt nun der Unterschied. In der amerikanischen Ausführung des Atari 400/800 ist nur die "kleinere" Version dieses Bausteins, der CTIA vorhanden. Dieser beschränkt die über BASIC ansprechbaren Graphikstufen auf 9 (0 bis 8).

Der GTIA Ihres Atari 400/800 kann mehr. Er übernimmt das Management für weitere drei Graphikstufen (9, 10, 11) und erlaubt Ihnen bis zu 16 verschiedene Farben, oder eine Farbe in 16 verschiedenen Helligkeitsstufen gleichzeitig darzustellen. Wie dies geschieht, wird in den folgenden Zeilen versucht zu erklären.

Zur Erinnerung und zur Vervollständigung nochmals eine Zusammenstellung der Farbbefehle für die Graphikstufen 0 bis 11.

Farbbefehle für die Graphikstufen 0 bis 8

Aus dem Buch haben wir gelernt, dass es zwei Befehle gibt, die zur Farbdarstellung benötigt werden. Mit dem SETCOLOR Befehl füllen wir einen "Farbeimer" mit der gewünschten Farbe und Helligkeit. Die erste Zahl des SETCOLOR Befehls (0 bis 4) bestimmt, welcher Eimer gefüllt werden soll. Die zweite Zahl (0 bis 15 entsprechend der Farbtafeln im Handbuch oder Tabelle GR. 11, bestimmt die Farbe. Die dritte Zahl legt die Farbhelligkeit fest (8 Stufen, 0 bis 14 in Zweierschritten).

Z. B. SETCOLOR 2, 3, 4

bedeutet der Eimer Nr. 2 wird mit der Farbe 3 (rot) in der Helligkeit 4 gefüllt.

Mit dem "COLOR" Befehl (z. B. COLOR 0 – 255) bestimmen Sie welcher "Pinsel" zum malen benützt wird. Jeder Pinsel gehört in den verschiedenen Graphikstufen zu einem bestimmten Eimer.

Vergessen Sie nicht: Um Farben darzustellen müssen Sie keine Farben benennen. Der Computer macht dies für Sie, indem er die Farbeimer (Farbregister) mit den Ausgangsfarben füllt. Sie können zu jeder Zeit diese Farben durch eine der 128 möglichen Farbhelligkeitskombinationen ersetzen. Welche Funktion die einzelnen Eimer und Pinsel in den verschiedenen Graphikstufen haben, ersehen Sie aus der folgenden Aufstellung. Denken Sie bitte daran, dass jeder neue Graphikbefehl die Farbeimer wieder mit den Ausgangsfarben füllt. Berücksichtigen Sie bitte auch, dass in den Graphikstufen 3 bis 7 Ihre "PLOT" und "DRAWTO" Befehle solange keine Wirkung zeigen, bis Sie die Ausgangslage (COLOR 0=gleiche Farbe wie Hintergrund) durch einen entsprechenden COLOR Befehl (1 grösser) geändert haben.

In den Graphikstufen 9 bis 11 stehen uns mehr Farben zur gleichzeitigen Darstellung zur Verfügung. Die Auflösung in diesen drei Graphikstufen beträgt horizontal 80 (0 – 79) Bildpunkte und vertikal 192 (0 – 191) Bildpunkte.

In der Graphikstufe 9 können wir eine Farbe in 16 verschiedenen Helligkeitsstufen darstellen. Die Helligkeitsstufen werden hier durch den COLOR Befehl (COLOR 0 bis 15 - dunkel bis hell) bestimmt. Die Farben des Hintergrundes bestimmen wir mit dem SETCOLOR Befehl für Farbeimer 4. In der Ausgangslage ist dieser mit der Farbe Schwarz gefüllt.

Versuchen wir es mit einem kleinen Programm.

```
10 GRAPHIK 9
20 SETCOLOR 4,3,0                (dunkelste Helligkeit von Rot)
30 FOR X = 0 TO 15
40 COLOR X                       (Helligkeit von 0 bis 15)
50 PLOT X,0:DRAWTO X,191
60 NEXT X
70 GOTO 70                       (WICHTIG, der Computer soll
                                 daran gehindert werden wieder in
                                 die Graphikstufe 0 zurückzu-
                                 springen).
RUN
```

Diese Befehle würden die entsprechenden Bildpunkte in verschiedenen Helligkeiten anschallten. Experimentieren Sie selbst. Geben Sie als letzten Befehl in Ihrem Versuchsprogramm stets folgendes ein: (Zeilennummer entsprechend Ihrem Programm) z. B. 70 GOTO 70. Tun Sie dies nicht, würde der Computer nach Ausführung automatisch in den Graphikmode 0 zurückspringen, den Bildschirm löschen und Sie könnten Ihr Kunstwerk nicht bewundern.

Im Graphikmode 10 stehen uns neun verschiedene Farben zur Verfügung. Die Auflösung ist die gleiche. Die Farbzuweisung geschieht hier ebenfalls mit dem COLOR Befehl, COLOR 4 bis 8 sind in dieser Reihenfolge mit folgenden Farben belegt: orangerot, gruen, blau, violetrot, schwarz. Mit den COLOR Befehlen 0 bis 3 rufen Sie jedes Mal die vom Computer gesetzte Grundfarbe schwarz auf. Die Farbeimer für die COLOR Befehle 4 bis 8 können Sie mit den bekannten SETCOLOR Befehlen (0 bis 4) ändern. Für die COLOR Befehle 0 bis 3 wird die Sache etwas komplizierter. Diese "Pinsel" tauchen in Eimer, die Sie nicht durch die gewohnten SETCOLOR Befehle erreichen können. Hier hilft uns nur der aus dem BASIC bekannte POKE Befehl. Mit

ihm können wir die Eimer füllen. Die Adressen lauten 704, 705, 706 und 707. Welche Zahl wir diesen Speicherstellen zuweisen müssen, errechnet sich nach folgender Formel: Farbnummer mal 16 + Helligkeitswert. Lassen Sie mich dies an einem Beispiel erklären.

Nehmen wir an Sie möchten den COLOR 1 Befehl statt schwarz die Farbe rot zuweisen. Die Adresse für den Farbeimer ist 705. Der Wert der dort abgelegt werden muss errechnet sich nun wie folgt:

Farbnummer für rot: 3
Helligkeitsstufe: 10
Farbnummer * 16 + Helligkeitsstufe ⟩ 3 * 16 + 10 = 58

Dies ist der Wert, der an der Adresse 705 abgelegt werden muß. Dazu müssen wir folgenden Befehl eingeben: POKE 705,58

Benützen wir nun COLOR 1 zum PLOTEN, so geschieht das in der Farbe rot in der Helligkeit 10.

Experimentieren Sie selbst.

In der Graphikstufe 11 können wir 16 Farben gleichzeitig zur Darstellung bringen. Die Auflösung ist die gleiche wie in den Graphikstufen 9 und 10.

Der COLOR Befehl (0 bis 15) spricht direkt die bekannten Farbnummern an. Die Helligkeit ist bei allen Farben gleich und wird durch den Helligkeitswert im Farbeimer 4 (SETCOLOR 4) bestimmt. Der Hintergrund ist in der Ausgangslage schwarz (Farbeimer 4 – SE.4,0,0). Die Hintergrundfarbe kann aber ebenfalls durch den Befehl SETCOLOR 4 geändert werden. Die Hintergrundhelligkeit wird vom Computer vorgegeben. Eine andere Farbe als schwarz verfälscht aber die Darstellung der restlichen Farben.

Dazu ein kleines Demoprogramm zum experimentieren.

```
5 FOR X=1 TO 2 STEP 0:REM IMMER WIEDER
10 GRAPHICS 11:REM NEUES BILD /GR.11
15 FOR Y=1 TO 50:REM 50 MAL
20 C=RND(0)*16:REM ZUFALLSZAHL >0UND <16
30 COLOR C:REM COLORBEFEHL RUNDET C AUF DEN
   GANZZAHLWERT AB (0-15)
40 PLOT 40,90:DRAWTO INT(RND(0)*80),INT(RND
  (0)*192):REM ZEICHENROUTINE
50 NEXT Y
55 FOR TI=1 TO 400:NEXT TI:REM ZEITSCHLEIFE
60 NEXT X
```

Übersichtstabelle – Farbbefehle in den Graphikstufen 0 bis 11

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFE 0 UND ALLE TEXTFENSTER

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
0	xxxxxxxxxxxxx	DER COLORWERT BE-	xxxxxxxxxxxxx
1	HELLBLAU	STIMMT HIER DAS	SCHRIFTHELLIGKEIT
2	DUNKELBLAU	ZEICHEN UND SEINE	HINTERGRUNDFARBE
3	xxxxxxxxxxxxx	FARBE.	xxxxxxxxxxxxx
4	SCHWARZ		RANDFARBE

----

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFE 1 UND GRAPHIKSTUFE 2 (TEXTMODI)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
0	ORANGE	DER COLORWERT BE-	ZEICHEN
1	HELLGRUEN	STIMMT HIER WELCHES	ZEICHEN
2	DUNKELBLAU	ZEICHEN GEPLOTTET	ZEICHEN
3	VIOLETROT	WIRD.	ZEICHEN
4	SCHWARZ		HINTERGRUND(s.RAND)

---

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFEN 3,5,7 (VIERFARBENMODI)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
0	ORANGE	COLOR 1	GRAPHIKPUNKT
1	HELLGRUEN	COLOR 2	GRAPHIKPUNKT
2	DUNKELBLAU	COLOR 3	GRAPHIKPUNKT
3	xxxxxxxxxxxx	xxxxxxx	xxxxxxxxxxxx
4	SCHWARZ	COLOR 0	GRAPHIKPUNKT UND AUSGANGSFARBE FUER: HINTERGRUND + RAND

---

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFEN 4 UND 6 (ZWEIFARBENMODI)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
0	ORANGE	COLOR 1	GRAPHIKPUNKT
1	xxxxxxxxx	xxxxxxx	xxxxxxxxx
2	xxxxxxxxx	xxxxxxx	xxxxxxxxx
3	xxxxxxxxx	xxxxxxx	xxxxxxxxx
4	SCHWARZ	COLOR 0	GRAPHIKPUNKT UND AUSGANGSFARBE FUER: HINTERGRUND + RAND

----

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFE 8 ...... (EINE FARBE UND 2 HELLIGKEITEN)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
0	xxxxxxxxx	xxxxxxx	xxxxxxxxx
1	HELLGRUEN	COLOR 1	GR.PUNKT-HELLIGKEIT
2	DUNKELBLAU	COLOR 0	GR.PUNKT/HINTERGRUND
3	xxxxxxxxx	xxxxxxx	xxxxxxxxx
4	SCHWARZ	xxxxxxx	RANDFARBE

----

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFE 9 (HORIZONTAL 80 VERTIKAL 192 . . . 16 HELLIGKEITSSTUFEN)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
4	SCHWARZ	COLOR 0	GR.P./HINTERGRUNDFARBE
xxxxxxx	xxxxxxxxx	COLOR 1	GR.PUNKT-HELLIGKEIT 1
xxxxxxx	xxxxxxxxx	COLOR 2	GR.PUNKT-HELLIGKEIT 2
xxxxxxx	xxxxxxxxx	COLOR 3	GR.PUNKT-HELLIGKEIT 3
xxxxxxx	xxxxxxxxx	COLOR 4	GR.PUNKT-HELLIGKEIT 4
xxxxxxx	xxxxxxxxx	COLOR 5-14	HELLIGKEIT 5 - 14
xxxxxxx	xxxxxxxxx	COLOR 15	GR.PUNKT-HELLIGKEIT 15

HELLIGKEIT 0(COLOR 0) HAT DIE GLEICHE HELLIGKEIT WIE DER HINTERGRUND.

---

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

GRAPHIKSTUFE 10 (HORIZONTAL 80 VERTIKAL 192 BILDPUNKTE . . . 9 FARBENMODE)

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

SETCOLOR (FARBEIMER) -bzw. POKE ADRESSE-	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
704	SCHWARZ	COLOR 0	GR.P./HINTERGRUNDFARBE
705	SCHWARZ	COLOR 1	GRAPHIKPUNKT
706	SCHWARZ	COLOR 2	GRAPHIKPUNKT
707	SCHWARZ	COLOR 3	GRAPHIKPUNKT
SETCOLOR 0 (708)	ORANGE	COLOR 4	GRAPHIKPUNKT
SETCOLOR 1 (709)	HELLGRUEN	COLOR 5	GRAPHIKPUNKT
SETCOLOR 2 (710)	MITTELBLAU	COLOR 6	GRAPHIKPUNKT
SETCOLOR 3 (711)	VIOLETROT	COLOR 7	GRAPHIKPUNKT
SETCOLOR 4 (712)	SCHWARZ	COLOR 8	GRAPHIKPUNKT

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

GRAPHIKSTUFE 11 (HORIZONTAL 80 VERTIKAL 192 BILDPUNKTE . . 16 FARBMODE)

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

SETCOLOR (FARBEIMER)	AUSGANGSFARBE	COLOR (PINSEL)	BEDEUTUNG UND ANWENDUNG
SETCOLOR 4 (s.Text)	SCHWARZ	COLOR 0	GR.P./HINTERGRUNDFARBE
XXXXXXXX	HELLORANGE (GOLD)	COLOR 1	GRAPHIKPUNKT
XXXXXXXX	ORANGEROT	COLOR 2	GRAPHIKPUNKT
XXXXXXXX	ROT	COLOR 3	GRAPHIKPUNKT
XXXXXXXX	VIOLET	COLOR 4	GRAPHIKPUNKT
XXXXXXXX	VIOLETBLAU	COLOR 5	GRAPHIKPUNKT
XXXXXXXX	VERSCH.BLAUTOENE	COLOR 6	GRAPHIKPUNKT
XXXXXXXX	VERSCH.BLAUTOENE	COLOR 7	GRAPHIKPUNKT
XXXXXXXX	VERSCH.BLAUTOENE	COLOR 8	GRAPHIKPUNKT
XXXXXXXX	VERSCH.BLAUTOENE	COLOR 9	GRAPHIKPUNKT
XXXXXXXX	GRUENBLAU	COLOR 10	GRAPHIKPUNKT
XXXXXXXX	VERSCH.GRUENTOENE	COLOR 11	GRAPHIKPUNKT
XXXXXXXX	VERSCH.GRUENTOENE	COLOR 12	GRAPHIKPUNKT
XXXXXXXX	VERSCH.GRUENTOENE	COLOR 13	GRAPHIKPUNKT
XXXXXXXX	BRAUNGRUEN	COLOR 14	GRAPHIKPUNKT
XXXXXXXX	HELLORANGE	COLOR 15	GRAPHIKPUNKT

FARBEIMER 4(SETCOLOR 4) ENTHAELT DIE HINTERGRUNDFARBE.SEIN HELLIGKEITSWERT BESTIMMT DIE HELLIGKEIT DER ANDEREN FARBEN.

==

NACHFOLGEND NOCH DREI WEITERE PROGRAMMBEISPIELE FUER DIE GRAPHIKSTUFEN 9-11.

```
0 REM COPYRIGHT ATARI
10 GRAPHICS 9
15 SETCOLOR 4,15,0
20 FOR Y=55 TO 0 STEP -10
30 FOR X=0 TO 24
40 C=X:IF X>11 THEN C=24-X
45 C=C+3
50 Z=Y+(X)
55 D=INT(SQR(144-(X-12)*(X-12)))/2
57 COLOR 15-C
58 PLOT Z,Y+7-D
60 DRAWTO Z,Y+7+D
70 COLOR C
80 DRAWTO Z,180-Y+D
180 NEXT X
190 NEXT Y
200 GO TO 200
```

```
0 REM COPYRIGHT ATARI
100 REM DEMO GR.10
115 DIM C(8):GRAPHICS 10:FOR Z=704 TO 712:READ R:R=R*16+8:C(Z-704)=R:POKE Z,R:NE
XT Z
116 DATA -.5,1,3,4,5,7,9,12,13
118 LIM=22:T2=3.14159*2/LIM:COL=3:E1=1:DIM D(LIM,2)
120 GOSUB 1500:FOR V=1 TO LIM:T=T+T2:GOSUB 1500:NEXT V
400 GOTO 1000
490 REG=705
500 FOR X=1 TO 8:POKE REG,C(X):REG=REG+1:IF REG>712 THEN REG=705
510 NEXT X:REG=REG+1:IF REG>712 THEN REG=705
520 POKE 77,0:GOTO 500
1000 REM
1005 FOR E=1 TO 10:E2=INT(E/2-0.5)
1010 FOR R=E1 TO E1+E2:CR=8-COL:IF CR=0 THEN CR=8
1015 V=0:COLOR CR:GOSUB 2000:PLOT X,Y
1020 FOR V=1 TO LIM:T=T+T2:GOSUB 2000:DRAWTO X,Y:IF V>=LIM/2 THEN COLOR COL
1025 NEXT V:NEXT R:COL=COL+1:IF COL=9 THEN COL=1
1030 E1=E1+INT(E/2+0.5):NEXT E
1200 GOTO 490
1500 D(V,1)=SIN(T):D(V,2)=COS(T):RETURN
2000 X=(30-R)*0.6*D(V,1)+40:Y=60*D(V,2)+80:RETURN
```

```
0 REM COPYRIGHT ATARI
10 REM DEMO GR. 9-11
80 ? "MODE (9-11)";:INPUT MODE:LIM=15:GRAPHICS MODE
90 IF MODE=9 THEN POKE 712,128:GOTO 115
95 IF MODE=11 THEN POKE 712,10:GOTO 115
100 FOR I=704 TO 712:READ R:POKE I,R*16+8:NEXT I:LIM=8
105 DATA -.5,12,13,14,15,1,2,3,4
115 FOR X=1 TO LIM
120 COLOR X:POKE 765,X:PLOT X*4+5,0:DRAWTO X*4+5,159:PLOT X*4+1,159:POSITION X*4
+1,0:XIO 18,#6,0,0,"S:":NEXT X
150 GOTO 150
```

INPUT OUTPUT – EINGABE AUSGABEN

Befehle und Medien

In diesem Kapitel sollen die Ein- und Ausgabeoperationen beschrieben werden. Aufbau, Anschluss und Bedienung der peripheren Geräte finden Sie in den dazugehörenden Handbüchern.

Der Dialog zwischen den einzelnen Medien erfolgt über das "zentrale Ein- und Ausgabe-Untersystem" (Central Input/Output [CIO] subsystem) kurz CIO genannt. Dieses ermöglicht dem Anwender den Dialog mit den Medien über eine einzige Schnittstelle und in einer von den Medien verhältnissmässig unabhängigen Weise zu gestalten.

In der Praxis bedeutet das folgendes: Der Ein- und Ausgabebefehl besteht aus zwei Teilen. Der erste Teil, der Medien unabhängige Teil, spezifiziert ob ein Lesebefehl oder ein Schreibbefehl vorliegt, und in welchem Format der Datenfluss erfolgen soll.

Der zweite Teil beinhaltet den Kennbuchstaben des anzusprechenden Mediums (z. B. "E: für Editor oder "P: für Drucker). Voraussetzung für den Dialog mit den Medien ist, dass sie durch einen "OPEN" Befehl für die Ansprache freigegeben werden und einen bestimmten Kanal (1–7) für den Datenfluss zugewiesen bekommen.

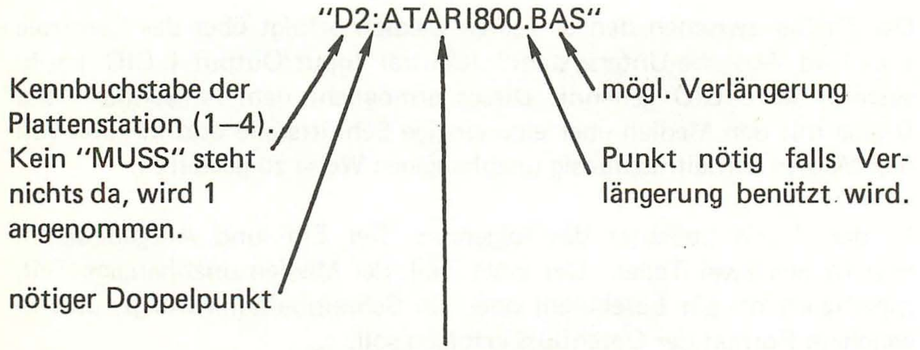
Dieser Kanal wird durch ein weiteres Untersystem IOCB (Input-Output Control Block) genannt, für den Dialog vorbereitet. Danach braucht man das entsprechende Medium (Recorder, Disk, Printer etc.) nur noch mit der Kanalnummer (#1 – 7) anzusprechen.

Die Befehle haben folgendes Format:

OPEN # A,B,C,D
CLOSE # A
A = Kanalnummer (1–7, das # Zeichen muß davor stehen)
B = Kennummer. Sie bestimmt die Art der Operation.
z. B. 4 = lesen
8 = ausgeben
12 = lesen und ausgeben
9 = Ausgabebefehl zum Ergänzen von Dateien.
Weitere Angaben siehe Basicreferencemanual.

C = Zusatzkennung (in den meisten Fällen null).
D = Kennbuchstabe des Mediums (z. B. "C:)

Eine Datei auf DISK (Magnetplatte) würde man folgenderweise ansprechen:



Dateiname (bis zu 8 Zeichen lang; muß mit einem Buchstaben anfangen.)

Anmerkung:
Mit dem Programmrecorder sind keine Dateinamen möglich.

Beispiel:

```
10 OPEN # 2,8,0, "D2:ATARI800.BAS"

oder

10 DIM A$(15):A$="D2:ATARI800.BAS"

20 OPEN # 2,8,A$

..........

90 CLOSE #2
```

Als Ausführungsbefehle stehen zur Verfügung: PUT GET PRINT INPUT (siehe auch Basicreference Manual)

Das ATARI Basicmodul kontrolliert 8 Bereiche im Arbeitsspeicher des

Computers (RAM), welche dem Betriebssystem die nötigen Informationen für die Ein- oder Ausgabeoperationen übermitteln. Zu diesen Informationen gehören der Befehl, Pufferlänge, Pufferadresse und zwei zusätzliche Kontrollvariablen. Die IOCB-Kanäle werden vom BASIC vorbereitet, aber der Anwender muß spezifizieren, welchen Kanal er für welche Medium benützen möchte.

IOCB #0 wird vom BASIC für den Schirmeditor reserviert. Ein Graphikbefehl reserviert IOCB#6 für die Ein- oder Ausgabe mit dem Bildschirm. (Gemeint ist hier das Graphikfenster "S:).

IOCB Kanal #7 wird vom BASIC für die Befehle LPRINT, CLOAD, und CSAVE benützt.

Kanal #0 darf nicht benützt werden. Sollten Sie Kanal #7 belegen, verhindert dies die Ausführung der auf diesem Kanal normalerweise durchgeführten Ein- und Ausgaben (z. B. LPRINT etc.). Die IOCB #1 bis #5 können vom Anwender nach eigenen Wünschen belegt werden.

("K:)	Keyboard	= Tastatur
("P:)	Line Printer	= Drucker
("C:)	Program-Recorder	= Kassettenrecorder
("D1–4:)	Disk Drive	= Plattenlaufwerke 1–4
("E:)	Screen Editor	= Texteditor
("S:)	TV Monitor	= Bildschirm
("R1–4:)	RS 232	= Serielle Schnittstelle (Modem 850)

Nach dieser reichlich theoretischen Betrachtung nun zur Praxis.
Der Dialog mit dem Kassettenrecorder.

Mit dem ATARI Kassettenrecorder können Sie Programme und Daten abspeichern und auch wieder in den Arbeitsspeicher (RAM) zurücklesen. Ausserdem dient er dazu, die auf der zweiten Spur mögliche Sprache oder Musik über den Fernsehmodulator des Computers wiedergeben.

Die Ausgabe oder das Lesen werden von Ihrem Computer durch Töne (Buzzer) angekündigt. (1 Ton für Lesen – PLAY Taste drücken und

dann "RETURN" oder 2 Töne für Ausgabe – PLAY und RECORD Taste drücken und dann "RETURN", Näheres finden Sie in Ihrem Handbuch für die Bedienung Ihres Cassettenrecorders.

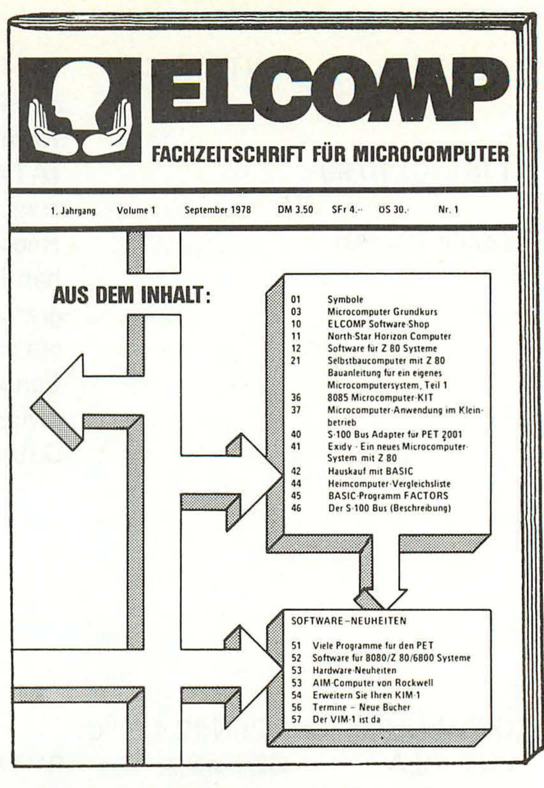
Es stehen folgende Befehle zur Verfügung:

Ausgabe Befehl	Lesebefehl	Datenformat	Anwendung
CSAVE	CLOAD	Daten werden in einem vom	Abspeichern und lesen
z.B. CSAVE	z.B. CLOAD	Basicmodul geschaffenen	von Programmen
od. 100 CSAVE	100 CLOAD	Zwischencode übersetzt und so platzsparend abgespeichert. Kurzer Abstand zwischen den einzelnen Datenblöcken.	
SAVE "C:	LOAD "C:	Wie oben, nur kurzer Abstand zwischen den Datenblöcken.	Abspeichern und Lesen von Programmen
	RUN"C:		Lesen und automatisches Starten eines Programmes Als letzte Zeile in einem Programm dient es zum Aneinanderreihen von Programmen oder Programmteilen.

LIST"C: LIST"C:,10,90 (Zeile 10–90)	ENTER "C:	Die Daten werden im Textformat (ATASCII) übertragen, d. h. der Rechner behandelt die Programmzeilen wie einen Text. Langer Abstand zwischen den Datenblöcken.	Abspeichern und Lesen von Programmen. Beim Lesen bleibt ein i. Arbeitsspeicher befindliches Progr. ungelöscht. Zeilen mit gl. Nummern werden aber vom neuen Programm überschrieben. Ergänzen von Programmen mit abgespeicherten Progr.-teilen. Oder Abspeichern eines noch nicht fehlerfreien Progr.
OPEN#2,8,0,"C:	OPEN#2,4,0,"C:		
PUT #2,A	GET #2,A	BYTEweise Übertragung	Zum Abspeichern und Lesen von Daten
PRINT #2,A$	INPUT #2,A$	Datenblockweise Übertragung	
A=numerischer Wert	A$=alphanumerischer Wert	(128 BYTE)	
(0–255)	(Zeichen)		

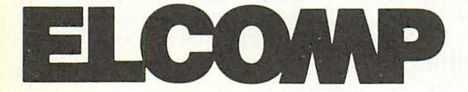
Auch Sie brauchen ELCOMP!





Jahresabonnement DM 59,-- incl. Mwst. und Versand.

Zurückliegende Hefte zu Originalpreisen noch verfügbar.



HOFACKER-VERLAG
Ing. W. Hofacker GmbH
Tegernseer Straße 18
D-8150 Holzkirchen/Obb.

Die Fachzeitschrift für MICROCOMPUTER
Eine unentbehrliche Informationsquelle für alle Elektroniker

Microcomputer-Anwendungsbeispiele
Künstliche Intelligenz
Block-Strukturierte Programme
Datenverarbeitung im Kleinbetrieb
Club-Neuheiten
Computer und Kunst
Musik mit dem Computer
Monitore für 8080, 6800, 6502, Z 80, SC/MP, 2650, 1802
Eigenbau-Computersysteme
Interface-Techniken
Microcomputer KITs
Neue Produkte
Betriebssysteme für Floppys
Programmiertechniken
Software-Quellen
Programmierbeispiele
Soziale Aspekte der Microcomputertechnik
Technologische Neuheiten
Anwendungen in der Meß- und Regeltechnik
Anwendungen bei Funk-Amateuren



–INHALTSVERZEICHNISSE

Inhaltsverzeichnis der zurückliegenden ELCOMP - Hefte

Ein einfaches PROM-Programmiergerät
Ein-/Ausgabeprogrammierung mit PET
Erweiterung für NASCOM-1
Lernen für die Anwendung von Microprocessoren

Nr. 9 September 4,50 DM
Tips für den BASIC-Programmierer
Wie lernt man BASIC, Teil X
EUROCOM-1
Michrochatter
Die neuen ROMs im CBM
Neue Publikationen
Speichererweiterung für den PET
Musikprogramm für den NASCOM-1
Computersystem 79
Umschalten des Bildschirms auf Breitschrift beim TRS-80 Level I
Vergleich dreier Zahlen
ELCOM-Leserbriefe
Diskettenpower APPLE II
Neuer Einplatinencomputer von Motorola
Einstellung der Tonspur beim PET

Nr. 10 Oktober 1979 4,50 DM
Ein-/Ausgabeprogrammierung mit dem TRS-80
Texas Instruments stellt seinen Heimcomputer vor
Microchatter
Wie lernt man BASIC, Teil XI
Der MZ-80 von Sharp
Microcomputer als Hobby
Ein komplettes Textverarbeitungssystem mit CBM-Computer u. Heathkit-Drucker
Wer ist OHIO SCIENTIFIC ?
KURDIS - Ein Programm zur Kurvendiskussion
BASIC im Unterricht
Monolithischer 8 Bit Analog/Digital-Wandler
Band-Test

Nr. 11/12 Nov./Dez. 1979 9,00 DM
Jetzt kommt PASCAL
Aus der PET-Modulkiste
HISTORGRAMM
32k RAM Zusatzspeicher f. PET 2001
Exidy Sorcerer
Kleinschrift und Verbesserung des Video-RAMs beim TRS-80
Filmkamerasteuerung mit PET
Anschluß eines Floppy-Disk-Controllers an 6500 Microprozessoren
Anschluß eines Baudot-Fernschreibers an einen Microcomputer
Ein Besuch bei ATARI
Kalender
Ein wenig Trigonometrie
Berechnen der Kreiszahl
Der NASCOM-2 ist da
PET als IC-Tester
Das NIM-Spiel und was es damit auf sich hat
Linking Loader für PET 2001 (8k)
MEMDMP-Hex Dump- Programm für 6502 Systeme

KIM-1 schreibt auf Skop
IRRGARTEN
KIM-Orgel
Lichtgriffel
Meldeton für PET
Preiswerter Drucker f. NASCOM-1
Wie lernt man BASIC, Teil XII

Bereits vergriffen !

Nr. 1 Januar 1980 4,50 DM
Microcomputer-Software und Vernunft
TRS-80 als "elektronisches Kurvenlineal"
Schrottknüppel
Programmbeschr.: FLAG OPERATION
Aus der PET-Modul-Kiste
Alternatives TINY-BASIC für den TRS-80 unter Level 1
Programmtricks APPLE II
NEU!!! Einführungskurs: Programmieren in Maschinensprache auf Z-80
Programmiertrick für TRS-80
Eine professionelle Analogschnittstelle
GRAPHSYS
Wie lernt man BASIC, Teil XIII
Programmierhilfen f. NASCOM-1
SUMA 85 (Selbstbausystem)
Automatische Zeilennummerierung für PET
ULTRA-MON-CBM Bedienungsanleitung
Entwicklungsboard f. AmZ8000
ELTRO-HOBBY 79, der große Renner

Nr. 2 Februar 1980 4,50 DM
"Echte" Großschrift beim TRS-80
BASIC-Programme - kürzer und schneller
Datenlogger BASIC - Programmierbar
EPROM-Programmiergerät für PET
HIGH-Resolution-Graphic auf dem APPLE II
Wie lernt man BASIC, Teil XIV
Der TI-Programmer
SUMA 85
Programmieren in Maschinsprache mit Z-80
PET-Petits

Nr. 3 März 1980 4,50 DM
Der Personalcomputer von HP ist da
Lernen mit dem AIM, Teil 1
Die neuen CBM-Modelle
SUMA 85, Teil 3
Programmieren in Maschinensprache mit Z-80, Teil 3
Maschinencodemonitor in TINY-BASIC
PET-Petits
Was ist Datenschutz?
Monitorprogramm für SC/MP
Wie lernt man BASIC, Teil XV
Quadratische Gleichungen
Wer ist Jim Butterfield?
Messebericht: Comsumer Electronics Show
Der PC-100

Nr. 4 April 1980 4,50 DM
Grundlagen der Finanzbuchhaltung für den Microcomputeranwender
SUMA 85, Teil 4
Messebericht: Hobbytronic Dortmund
Berechnung ganzrationaler Funktionen
Aus der PET-Modulkiste
SDS-100 MicroMini
Erstes Home-Computer-System von TI
Programmieren in Maschinensprache mit Z-80, Teil 4
PRINT USING für X80 und PET
Microchatter
Wie lernt man BASIC, Teil XVI
PET Petits
Grafik mit dem Superboard
Untersuchen von Texten auf die Häufigkeit des Auftretens einzelner Buchstaben
Zum lernen ideal. . . .
Hochinteressante Interviews

Nr. 5 Mai 1980 4,50 DM
Aus der PET-Modul-Kiste
Z80-Disassembler für den NASCOM
Implementierung einer REAL-TIME-CLOCK
SUMA 85, Teil 5
Programmieren in Maschinensprache mit Z-80, Teil 5
Lernen mit AIM, Teil II
AIM 65 hilft im Büro
Protokollieren von TRS-80 Bildschirmausgaben
Wie lernt man BASIC, Teil XVII
West Coast Computer Faire
Der HY Q-1000
Der $BASE_2$ Drucker

Nr. 6 Juni 1980 4,50 DM
Die einfache Verkettungstechnik
Programmieren in Maschinensprache mit Z-80, Teil 6
Lernen mit dem AIM, Teil 3
Microcomputer Fools Corner
Challenger C4PMF Microcomputer
SUMA 85, Teil 6
Wie lernt man BASIC, Teil XVIII
TINY-FORTH für TRS-80 Level II, 16k
CBM-Disk BASIC Version 4.0
Heathkit - Matrixdrucker / Erfahrungsbericht
Monostabile Kippstufen
5. West Coast Computer Faire
Hex-DUMP in BASIC für Superboard
Michrochatter

ELCOMP ist in der Microcomputer-Berichterstattung immer vorne.
Keiner, der mitreden will, kann es sich noch leisten, ELCOMP nicht zu lesen! Bestellen Sie deshalb heute alle noch verfügbaren Hefte und sichern Sie sich ELCOMP im Abonnement für die Zukunft.

Hiermit gebe ich folgende Bestellung unter Anerkennung
Ihrer Lieferungs- und Zahlungsbedingungen auf.



Menge	Artikelbezeichnung	Einzelpreis

Menge	Artikelbezeichnung	Einzelpreis

Unterschrift (für Jugendliche unter 18 J. der Erziehungsberechtigte) Datum

---

BESTELLKARTE für ein ELCOMP-Abonnement

Auslands-Abonnement (Europa) + DM 10.-- Versandkosten

- O Bitte senden Sie mir ab sofort alle künftigen Ausgaben der ELCOMP. Preis DM 59,-- pro Jahr incl. Versand und Verpackung. Lieferung ab Ausgabe.............................
- O Bitte senden Sie mir alle bereits erschienen Ausgaben zum Sonderpreis DM 37,-- (Sept. 78 – Okt. 79, außer Nr.3/4 1978, Nr. 2, 4,11/12 1979)
- O Jahrgang 1980 DM 49,–
- O Ich bezahle im voraus für..........Jahr(e) DM.................., Pschk. München 15994807, Dresdner Bank München Kto. 5390 370, BLZ 700 800 00.
- O Ich bitte um Lieferung auf Rechnung.
- O Ich bitte um Abbuchung von meinem Konto (Giro od. Postscheckkonto)
  Kto.Nr.. BLZ...
  Geldinstitut:.................................... Ort:..

Name/Vorname

Straße / Hs-Nr.

PLZ Ort

Datum.. Unterschrift..

ELCOMP

Absender
Bitte deutlich ausfüllen

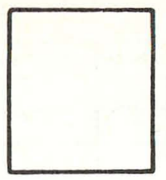
____
Vorname/Name

____
Beruf

____
Straße/Nr.

____
Plz Ort

POSTKARTE



ELCOMP

MIKROCOMPUTER BOOK STORE

Tegernseerstr. 18

D-8150 Holzkirchen /Obb.

ABSENDER:

....
Name, Vorname

....
Straße

( ) ....
PLZ Ort

....
Telefon

Das Abonnement wird automatisch verlängert
(Kündigung 8 Wochen z. Abonnement-Ablauf)



Ing. W. Hofacker GmbH
Tegernseer Straße 18
D-8150 Holzkirchen


ATARI 400
CONTROLLER JACKS


ATARI 800